POLITECNICO DI MILANO

Scuola di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Ingegneria dell'Automazione

ALGORITHMS FOR POINT CLOUD ELABORATION AND 3D RECONSTRUCTION OF YACHT SAILS DURING NAVIGATION

Relatore: Prof. Alfredo CIGADA
Correlatore: Ing. Ambra VANDONE

Master Thesis by:

Eugenio CANCIANI

Matriculation Number: 799460

A.A. 2013-2014

ABSTRACT

This work presents a procedure for the acquisition and the 3D reconstruction of the sail shapes during navigation. Large differences, in fact, exist between a computer based design shape and the resulting flying shape due to different factors such as pressure distribution, sail trim controls and fluid structure interaction forces. Nowadays Computational Fluid Dynamics (CFD) codes assess the yacht performances starting from the design shape; however, the actual flying shape is the only one truly related to moments, heeling and thrust forces. Hence the reason for this study. Moreover, analyzing the reconstructed shape in terms of geometrical features can be of interest for sail makers to redesign the sail and for crew members to adjust the trim. The main goal of this work was the development of a methodology for data acquisition and elaboration. A Time-Of-Flight 3D based device was realized ensuring non-contact, wide range and outdoor measurement. Its metrological qualification was performed, including tests to assess the influence of distance, incident angle, target material and lighting conditions. Furthermore, most efforts were addressed to the development of a custom post process algorithm. Raw data are acquired in terms of point cloud and several steps are required to lead to the sail surface reconstruction. Those are primarily: registration of the different scans into a common reference system, scene interpretation - i.e. segmentation of the cloud to extract the sail cluster -, filtering of the data to remove outliers and to reduce measurement uncertainty, and meshing - i.e. the creation of a surface -. The procedure was validated onto synthetic data sets representing simple scenes and onto design sail shapes. Finally, the algorithm was exploited to reconstruct sails during wind tunnel campaigns and even for few tests on field leading to promising results.

SOMMARIO

Col presente lavoro si descrive lo sviluppo di una metodologia volta all'acquisizione e alla ricostruzione della forma delle vele durante la navigazione. Esistono infatti, notevoli differenze tra la forma della vela disegnata dal costruttore e la forma che essa assume sotto l'azione di diversi fattori quali: distribuzione delle pressioni, regolazioni delle vele e forze generate nell'interazione fluido struttura. Attualmente, le prestazioni di un'imbarcazione a vela vengono valutate grazie a analisi CFD includendo nel modello le vele di progetto, tuttavia è la forma reale della vela in navigazione la vera responsabile di sbandamento e forze di propulsione. Ecco perché, la conoscenza della forma reale può risultare di particolare interesse in campo nautico. Inoltre, un'analisi geometrica di tale forma può essere utile ai velai per eventuali modifiche in fase di design e agli equipaggi per migliorare le regolazioni.

L'obiettivo principale è, quindi, lo sviluppo di una metodologia per l'acquisizione ed elaborazione dei dati. Uno strumento di acquisizione basato sulla tecnologia laser a tempo di volo è stato realizzato, garantendo misure 3D, senza contatto, ad ampia regione di indagine, sia indoor che outdoor. La qualificazione metrologica dello strumento è stata effettuata e ha permesso di valutare l'influenza sulla misura di diversi fattori quali distanza dal sensore, angolo di incidenza del raggio laser, materiale del bersaglio e luminosità ambientale. La maggior parte degli sforzi sono stati rivolti allo sviluppo del software dedicato di post processing dei dati che vengono acquisiti sotto forma di nuvole di punti. Diversi passaggi sono necessari per giungere alla ricostruzione 3D della superficie velica. In particolare, sono: registrazione di diverse scansioni in un comune sistema di riferimento, interpretazione della scena – cioè segmentazione della nuvola al fine di estrarre il cluster della vela -, filtraggio dei dati per eliminare gli outliers, e creazione della superficie. L'algoritmo è stato validato attraverso test su dati sintetici, rappresentanti scene semplici, e su dati rappresentanti la forma di progetto. Il software è quindi stato utilizzato per la ricostruzione di vele, sia acquisite in galleria del vento, sia in navigazione; restituendo risultati sono soddisfacenti.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 The Research Context

Computational Fluid Dynamics (CFD) analysis has become a valuable tool in yacht design. Compared to wind tunnel or towing tank testing it is a cost efficient tool for analyzing aero- and hydrodynamics of design variations, allowing to alter geometry systematically according to predicted forces. While CFD analysis has become routine in hull design for the investigation of hydrodynamic forces, its application in predicting the aerodynamics of a yacht is not yet widely exploited since modelling what happens above the waterline is always a complex work. The angles of attack of the onset flow vary greatly and regions of flow separation are inevitable. Moreover, the geometry of the sails is usually unknown. In fact, the design shape is just one special case of the set of possible flying shapes that a sail might assume. The aerodynamic pressure distribution acting on the surface of the sail, the forces resulting from rig design and trim, and the elasticity of the sail material influence its actual flying shape in varying wind conditions. The combination of all these factors leads to a virtually infinite number of flying shapes and, this is especially true considering offwind sails because of the lightweight construction materials and their relatively unconstrained nature. On the other hand, the geometry of the sails determine the aerodynamic flow around them and the resulting forces. It is thus of fundamental importance for the reliability of sail force predictions using CFD to know the real geometry of the sail at given wind conditions.

It is possible, although computationally expensive, to model the areo-structural coupling by combining CFD codes with Finite Element Analysis (FEA) of the

sail. Up to date numerical codes still need a massive validation work on both wind tunnel tests as well as on water testing at full scale. Direct acquisitions can give a substantial contribution, supposing that reliable flying shapes are provided, reflective of the realistic sailing and trim conditions.

1.2 The State Of The Art

Several contributions can be found in literature aiming at assessing sail flying shapes. One of the first attempts in measuring the resulting shape variation of a design shape, focusing on downwind sail in different wind conditions using wind tunnel tests, has been made by Razenbach and Kleene in [Ranzenbach R., 2002]. The resulting shapes were found using a Coordinate Measuring Machine (CMM) and a photogrammetry based technique. Photogrammetry has also been used at Politecnico di Milano Twisted Flow Wind Tunnel by Fossati et al. [Fossati et al., 2008] where an in house near IR-camera camera measurement system has been developed to recover sail flying shapes during wind tunnel tests. The photogrammetric technique has been exploited also at the YRU-Kiel Wind Tunnel, and in [Graf and Müller, 2009] Graf and Muller presented some tests on a set of spinnakers for an IMS600 custom design, comparing flying and design shapes, wind velocity and wind twist impact on flying shape. More recently, in [Renzsch and Graf, 2011], results of wind tunnel tests on two different asymmetric spinnakers have been reported. As a general comment, it can be said that the photogrammetry based technique was judged accurate and relatively fast during the tunnel occupancy phase, requiring only that digital images were recorded from at least three vantage points. Its chief disadvantages are that it requires an intensive data post-processing and that suffers from occlusion problem. This means that sometimes some grid points are not properly recorded by the cameras. To overcome the problem, a large number of cameras is required, leading to relevant difficulties in system set up in the wind tunnel. On the other hand, CMM processes requires a longer period during the tunnel occupancy phase (about half an hour per configuration), but yields results within moments after the digitizing process is complete. Its principal disadvantage is that the CMM and its operator can potentially influence the shape of the sail, moreover it requires the user to accurately select points directly upon the surface of the sail with the digitizing arm stylus. As far as the flying shape recovering at full scale are concerned, a valuable research activity concerning sail shapes and perfor-

mance measurements at full scale using a sail dynamometer boat called Fujin has been presented in [Masuyama and Fukasawa, 1997], [Masuyama et al., 2009] and [Fossati et al., 2008]. The sail shape was recorded using pairs of CCD (Charge-Coupled Device) cameras: horizontal stripes were drawn on the mainsail and jib at different heights and the image processing software retrieved shape parameters at these sail sections, such as maximum camber, maximum draft and twist angle values. The same procedure is used by several computer programs, which are nowadays commonly available on the market aiming at analyzing sail pictures taken onboard by means of a standard camera and providing some sail shape parameters in a certain number of sail sections. For instance, North Sails has developed a proprietary software called ASA (Advanced Sail Analyzer) to digitize pictures deriving synthetic parameters for each section that has been marked by the horizontal stripes on the sails. Upwind sails aerodynamics with flying shapes measurements at full scale have been also provided by the 33-foot dynamometer boat DYNA ([Hochkirch and Brandt, 1999]): for that project a system based on photogrammetric methods was used. The hardware, which mainly consists of a set of six digital cameras installed in fixed position on the boat, is described in details in [Clauss and Heisen, 2000] and a grid of discrete markers applied to the sails forming a grid of horizontal and vertical lines is used to define their flying shape. More recently, in [Le Pelley and Modral, 2008], a method called Visual Sail Position and Rig Shape (V-SPARS) aimed at measuring also downwind sails has been presented. It is based on cameras that capture fluorescent colored stripes on the sail and on an image processing software that produces the global coordinates of each stripe relative to a fixed datum position on the yacht. This method has been used for downwind sail aerodynamics investigation at full scale combining pressure and sail shape measurements ([Motta et al., 2014], [Deparday et al.,]). The above mentioned background is extremely useful not only for use while racing but also for sail design tool development which rely on a trustworthy validation process. One of the main drawback is that these systems can recover only few horizontal sections, deriving the relative angle between them but not being able to describe the three-dimensional position of the leading edge, therefore not being able to reproduce with accuracy the entire sail surfaces, which represent respectively the reference or the input in case of FSI or CFD calculations. Moreover, these systems rely on the assumption that the stripes remain in a horizontal plane, which is questionable for downwind sails especially when a large range of apparent wind angle is considered.

Considering the above, we decided to develop a new tool for the sail shape acquisition based on Time Of Flight (TOF) technology. It allows us to perform three dimensional (3D) measurement of the entire sail shape, overcoming the limits of the techniques mentioned above. Moreover, it is suitable both for indoor and outdoor environment as it is not sensible to light conditions. It presents a wide measurement range that perfectly meets the nautical field application.

1.3 Aims And Objectives

Aware of the importance of recovering the actual flying sail shape to assess the yacht performances, as explained so far, the current work presents an innovative tool developed within the Lecco Innovation Hub Sailing Yacht Lab Project ([Fossati et al., 2013]). It concerns the realization of a new generation sail dynamometer boat fitted with instruments dedicated to the acquisition of data on the behavioral variables of boat and its components at full scale, to support a scientific approach to research activities related to sailing yachts design and performance. To retrieve the sail shape data in dynamic situations a dedicated measurement system has been developed, as well as a proper data elaboration procedure. In fact, the tool considered starts from the acquisition of the geometrical information regarding the sail shape, assumed in a specific time instant and ends with the reconstruction of a 3D surface.

More in detail, data acquisition is performed through a LABView dedicated software retrieving point spatial coordinates of the sampled scene. The elaboration algorithms exploits all open source libraries. Code is implemented using C++ programming language and enclosed in a Qt Creator application. Qt Creator is a cross-platform C++, JavaScript and QML integrated, open-source, development environment, which is part of the SDK for the Qt GUI Application development framework. The scripts run on a Linux operated system, in order to facilitate the integration with external libraries. Qt Creator supplies the user interface part of the software, while operations on data points such as visualization, filtering and meshing are based on Point Cloud Library (PCL). PCL is a large scale, open project for 2D/3D image and point cloud processing. Computational Geometry Algorithms Library (CGAL) is exploited too. Once more, the CGAL Open Source Project is a library aimed to provide easy access to efficient and reliable geometric algorithms in the form of a C++ library.

All the elaboration procedure steps are described in detailed in the following chapters.

In particular, the structure of this theses counts:

- in Chapter 2, a brief description of non-contact measurements technique to motivate the choice of the technology we considered to test, the description of the acquisition unit and its metrological qualification;
- in Chapter 3, a discussion of all the data post processing steps such as data registration in a reference coordinate system, the scene interpretation and the extraction of the sail cluster, noise filtering, meshing and sail surface reconstruction. For each steps, the most common approaches presented in literature are discussed and combined or modified to suit our application. Once the strategy is selected, results for tests on synthetic data and on real acquisitions are reported;
- in Chapter 4, two study cases to present the algorithm performances on wind tunnel and on field acquisitions;

Finally, in the Conclusion paragraph, our consideration on the developed work and possible ideas for further improvements, such as making the algorithm more robust and automated, are reported.

Chapter 2

The Acquisition System

Measuring a shape consists in its sampling by using a proper technology. In case of regular geometry shapes fewer points need to be acquired, because, since the shape is a priori known, just its dimensions have to be measured. Instead, free-form shapes require a large number of points in order to get a proper reconstruction, consequently the measurement process must be optimized in terms not only of accuracy but also of speed. A point cloud data set is created through a 3D scanner or a depth camera; these devices register the external surfaces of an object or scene and for each sample return the information needed to retrieve its spacial coordinates. Basically there exist two families of methods allowing user to measure a shape: one of them requires the contact between a probe and the surface to be sampled (contact techniques), the other one does not (non-contact techniques).

2.1 Contact Measurement Techniques

Nowadays almost all the industrial metrology applications requiring very high accuracy are handled by using computer controlled Coordinate Measurement Machine (CMM) [Huang et al., 2004]. Basically, a probe is mounted on a traversing frame with three orthogonal axes embodying a spatial coordinate system and the points are sampled using that probe, whose position is known with very high accuracy in the coordinate system. In spite of its high measure performances, this technique has two main drawbacks: it is time-consuming, because just one point is sampled at a time, and it introduces intolerable load effects dealing with deformable shapes. As this research is aimed to classify sail shapes, this technique cannot be used.

2.2 Non-Contact Measurement Techniques

Between the non-contact optical measurement techniques able to get a complete 3D information [Curless, 1999], [Trucco and Verri, 1998], only four of them may be considered suitable to measure quickly and reliably big shapes: stereo vision, active triangulation and Moiré and Time-of-Flight (TOF) scanners.

2.2.1 Stereo Vision

Optical passive measurement systems determine the 3D scene coordinates by using the information contained in at least two images acquired synchronously. This approach uses the principle of the triangulation: two images of the same point P, acquired by two calibrated cameras placed in well-known positions, contain all the information required to compute the 3D coordinates of P in the reference system of one camera. Based on the known geometry of the vision system (see Figure 2.1), each camera is able to find out the direction of P. This direction corresponds to infinite points lying on a straight line passing through the camera optical center. The second camera, suitably angled, singles out a second straight line that has to contain the point P. Evidently the intersection of the two straight lines coincides with the point P.

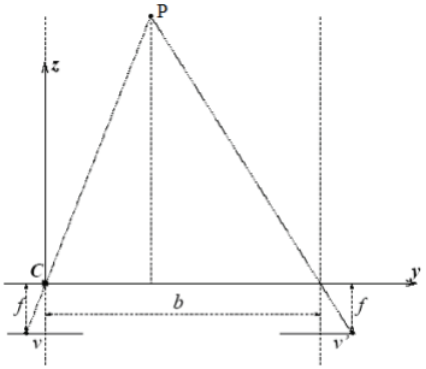


Figure 2.1: Principle of triangulation in stereo vision

The main difficulty of the method is the identification of the homologous points in the different images. The homologous points are the points identifying the same point in the scene. These landmarks can coincide with points easy to be identified, like edges or corners, or they are "artificially created" by using markers or projecting a laser spot (in this case this method is called active stereo vision). In literature

several algorithms that reconstruct the 3D by using a suitable number of homologous points [Kobayashi et al., 1990], [Molton et al., 1998] are presented. The use of this technique with markers is for example widespread in medical applications [Hajeer et al., 2001]. Moreover another issue is due to the fact that real images do introduce considerable uncertainties in the determination of the homologous points. Thus it is necessary to use error minimization techniques in order to get the 3D coordinates. The stereo vision is considered as an intrinsically dense method, because it allows user to get 3D information without using images shot at different times.

2.2.2 Active Triangulation

In this method the triangulation is performed intersecting the straight line containing the point to be measured identified by a digital camera with a light pattern (very often a plane) that illuminate that point. In a way the light pattern takes the place of the second camera.

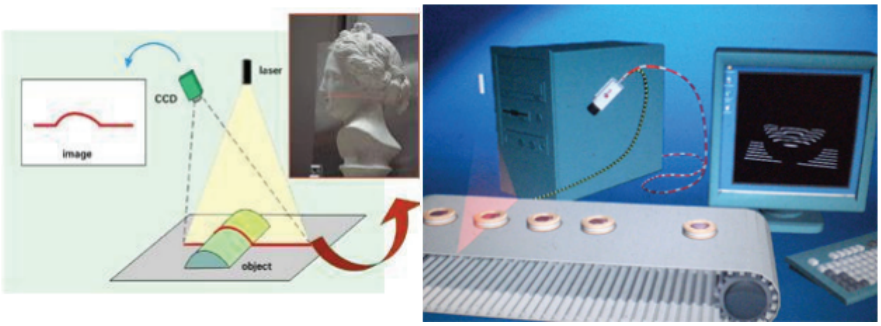


Figure 2.2: Principle of active triangulation (left) – model of scan in progress (right, courtesy of SICK)

It follows that the camera is able to "see" only the points illuminated by the pattern, so this method is not intrinsically dense. If a plane is projected, the intersection with the scene is a curve, called profile. In order to measure a whole object, several profiles must be acquired, so the measurement device has to be moved. Usually the resolution is not homogeneous: its value along the scan lines is generally much higher than between the lines. It is possible to increase the number of the acquired profiles, but it means to increase the acquisition time and, above all, the data processing time; this choice is a trade-off. The correspondence between the world and the image processing is obtained through the device calibration: it is possible either to calibrate the camera or to calibrate the whole measurement

system. Its sensibility and its measurement range are related to the relative positioning between the camera and the light pattern: an improvement of the sensibility makes the measurement range smaller and vice versa. The device able to measure the profile coordinates is called profilometer: it is made up of a camera, a device that generates the light pattern and a support. Usually the light pattern is a plane generated using a laser diode. Although a laser diode is fairly expensive, lasers have very good qualities: the diodes are small and do not have problems related to the heating, the light can be properly collimated and, since it is narrow-banded, the noise due to light present in the scene can be dramatically reduced mounting on the optics an interferential filter properly dimensioned. The complete measurement device, made up of the profilometer and the motion system, is called scanner. In a first stage the profile coordinates are expressed in the profilometer reference system (it depends on the calibration procedure), then all the profile coordinates are related to each other provided that the position of the profilometer when each profile has been acquired is well-known. A profilometer can be translated, rotated or even rototranslated, in accordance with the particular needs.

2.2.3 Moiré Interference Fringes

Moiré does not identify a well-defined technique, instead several methodologies which share the physical principle: the interference fringe analysis [Patorski, 1993]. When two grids, made up of equispaced bright and dark stripes are superimposed, dephase, because of the resulting mechanical interference, some interference fringes, that lie on the surface, arise. Practically, an undistorted grid is projected and acquired by using a digital camera, then the same grid is projected onto the object to be measured. This time the grid is distorted and its phase modulation is related to the measured object depth: this deformed grid id acquired too. The comparison of the phases allows the user to find out the depth coordinate; the other two coordinates are computed by means of the camera calibration. Basically there exist two families of Moiré methodologies [Harthong et al., 1991]:

- Projection Moiré: a grid is projected onto the surface of the object to be measured; the interference is generated by looking at the scene through a second grid equal to the first one, but slightly rotated [Takeda and Mutoh, 1983].
- Shadow Moiré: just one grid is used and it is placed close to the object to be measured, which is illuminated by using a punctual source. There is a

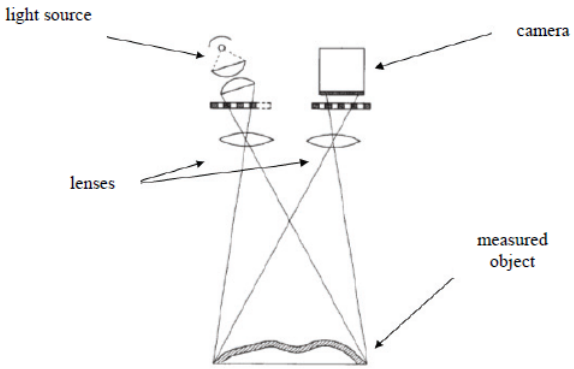


Figure 2.3: Projection Moirè functioning scheme

slight parallax between the light source and the user and the grid shadow on the object interferes with the grid itself: some interference fringes arise [D'Acquisto et al., 2002], [Dursun et al., 2003].

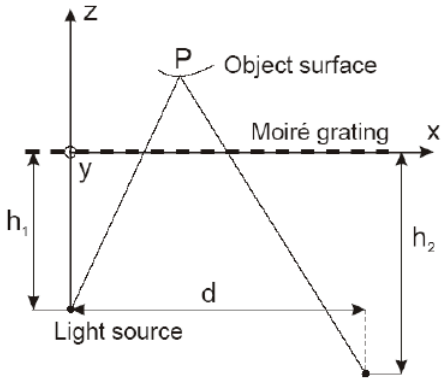


Figure 2.4: Shadow Moirè functioning scheme

In general Moiré methods are relative methods and are more suitable for continuous surfaces, without sudden differences in depth, sharp edges or high curvature regions, because they would generate phase differences difficult to be evaluated. Like stereo vision, these techniques are intrinsically dense. Basically the analysis follows the scheme represented in Figure 2.5.

The phase unwrap is easy to be carried out if the phase variation is less than 2π, otherwise an ambiguity in the phase value arises and this has to be tackled by using some sophisticated algorithms. For that reason, Moirè methods are more effective dealing with continuous and low curvature objects. The use of the Direct and the Inverse Fourier Transform and the filtering operation in the middle is aimed

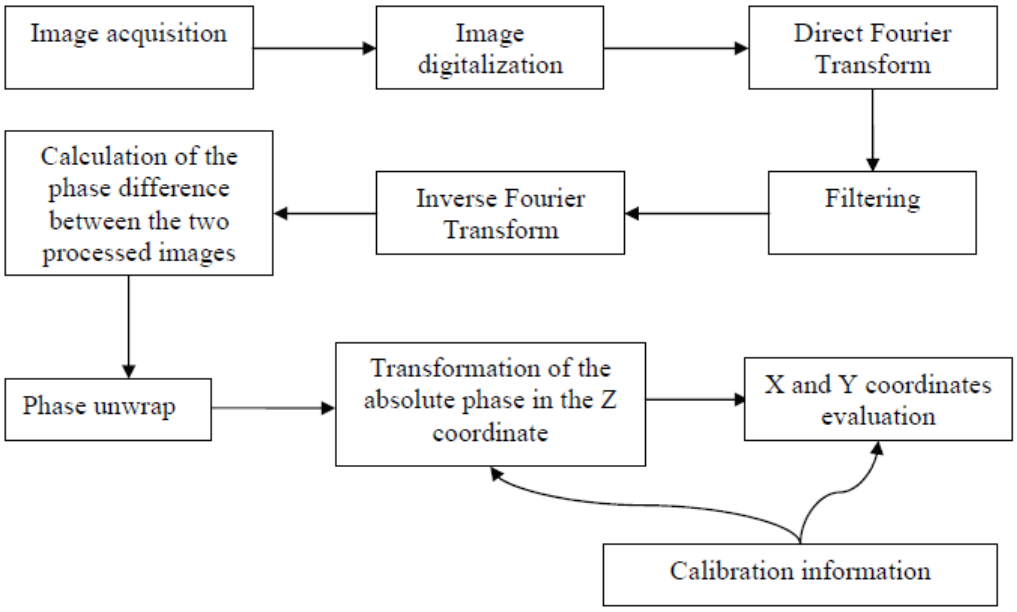


Figure 2.5: Moirè Analysis Pipeline

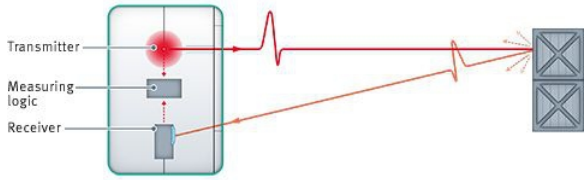
to discriminate the amplitude and the phase modulation, because only the latter is related to the depth of the measured object. Finally it is worth recalling that the Moirè are relative measurement methods, then the shape is in general reconstructed up to a scale factor.

2.2.4 Time-of-Flight (TOF)

The acquisition techniques explained are generally utilized for registering small volumes, up to human figures, while for bigger shapes a different principle is used, due to the divergence of the light patterns or to the sensor dimension requested in the active triangulation case. For bigger targets, usually, the acquisition units exploit the Time-of-Flight principle, which is essentially retrieving the time needed by a light emitted to travel to the target and then return to the measurement system [Guidi et al., 2010]. The knowledge of the time, along with the properties of the electromagnetic wave used, allows to go back to the distance between the instrument and the surface. Although the measure retrieved is a punctual value, adding additional information on the tilt and span angle allows to have to all the three dimensional coordinates for every point. Systems that utilize the principle of the distance measure are usually referred as Laser Scanners o LIDAR (LIght Detection And Ranging). There are two ways to measure the time of flight of the

wave:

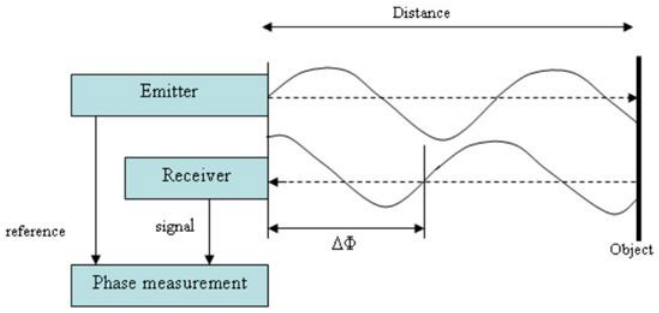
- Direct (pulsed) time-of-flight: the time measured is obtain with a single pulse of light, which is shot towards the target surface, while a timer records the instant of start and return. The time taken to travel and return is given by the subtraction of these values.



$$d = \frac{\Delta T * c}{2}$$

Figure 2.6: Pulsed Time-of-flight principle

- Indirect (continuous wave) time-of-flight: the light emitted is a continuous, amplitude modulated (AM) wave. The beat from the reference wave and the returning, shifted, wave returns the information on the time searched, as it is proportional to the shift between the two signals. When the shift exceeds the oscillation cycle of the wave, an ambiguity is introduced, which can't be eliminated without the introduction of additional information.



$$d = \frac{c * \varepsilon}{4 * \pi * f_0}$$

Figure 2.7: Continuous Wave Time-of-Flight principle - ε is the beat signal: $\varepsilon = s(t + T_0) - s(t)$ = signal received - signal emitted

2.3 Choice Of The Technology

The research purpose is to analyze big free-form shapes, as are the sails, both in a indoor, controlled environment, the wind tunnel and during output navigation. The acquisitions have to be reliable and quite quick in many cases, due to the flap in some sailing trim. Moiré interference fringes analysis is characterized by several drawbacks [Marshall et al., 1998]: it is not reliable dealing with high curvature surfaces and it is very difficult and expensive to get a projector generating high quality big grids, above all in an industrial environment. Moreover calibration procedure is very complex and it is very complicated to realize during the measurement if something is wrong. At last, analysis algorithms are not trivial and the analyzed signals have to satisfy the periodicity conditions required by the Fourier transform [Bendat and Piersol, 2011]: these signals also introduce problems like leakage that does not exist using the other techniques. Stereo vision is currently used in others works [Salzmann et al., 2007] to retrieve the sail shape, however its use is not effective in the outdoor campaign, because of the presence of strong ambient light, which interferes with the markers detection. The same issue is present also in the active triangulation technique and, in general, in every choice involving cameras, as they are likely to saturate in the outdoor acquisitions. This is the main reason that lead to the choice of a Time-of-Flight acquisition system, which is mainly unaffected by the environment conditions. Moreover the laser scanners are able to detect without problems large objects, which is the case, with sail up to 12/15 meters high. To avoid misinterpretation due to ambiguities, the direct time technique is preferred. In the following the measurement device that has been chosen and integrated is described and characterized.

2.3.1 The Sick LMS 511 Laser Scanner

The Sick LMS 511 is a laser scanner based on time-of-flight (TOF) technology. The operating principle is widely described both in [Reina and Gonzales, 1997] and in [Guidi et al., 2010], while a schematic representation is reported in Figure 2.8: a pulsed infrared laser beam at 905 nm is emitted and reflected on the target surface back to the sensor.

The time between the transmission and the reception of the laser beam is used to measure the distance between the scanner and the object. The sensor scans the surrounding perimeter on a plane, thanks to a rotating mirror that deflects the laser

beam, so the position of the object is given in the form of distance and angle, as in Figure 2.9.

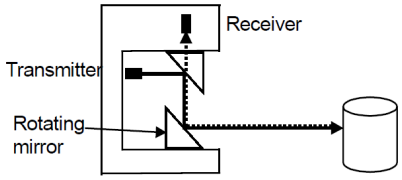


Figure 2.8: SICK Operating Principle Scheme

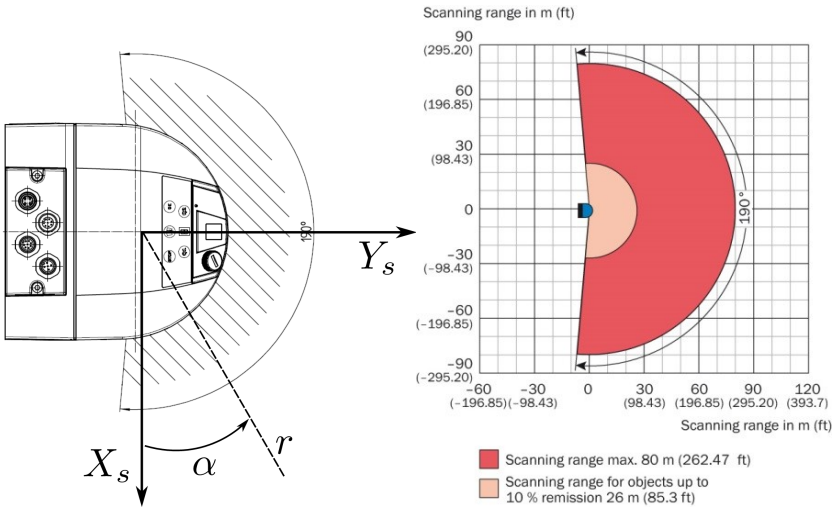


Figure 2.9: SICK Scanning Principle Scheme and Range of Measurement

The LMS 511 operates with many possible scanning frequencies (25Hz / 35Hz / 50Hz / 75Hz / 100Hz), therefore achieving different angular steps for every configuration. Since the field of view is 190°, with a starting angle of -5° and a stop angle of 185°, the angular resolutions selectable are 0.167° , 0.25° , 0.333° , 0.5° , 0.667° and 1°. We selected a value of 0.5° and fixed it for all the tests. Thus, each scan is composed of 381 measured ranged points. According to the manufacturer's specifications [Sheet, 2014], the scanner can measure ranges up to 80 m for 100% of object remission with accuracy of ±12 mm at a distance of 6 m. For a 10% target remission, the device presents for distances of 1 to 10 m a nominal systematic error ± 25 mm and a statistical error ± 7 mm; for distances 10 m to 20 m the nominal systematic error is ± 35 mm and the statistical error ± 9 mm. The LMS 511 itself can scan only a plane, following the prism's rotation; to obtain

a full three-dimensional acquisition an further movement, the tilt rotation, has to be added. A rotation around a horizontal axes has been implemented screwing the SICK unit to a metal plate, which is in turn connected to a servomotor. Both the information from the laser scanner and from the servomotor are used in a dedicated LabView software to retrieve the spacial coordinates of the point considered. The full explanation of this work is object of [Perego, 2008].

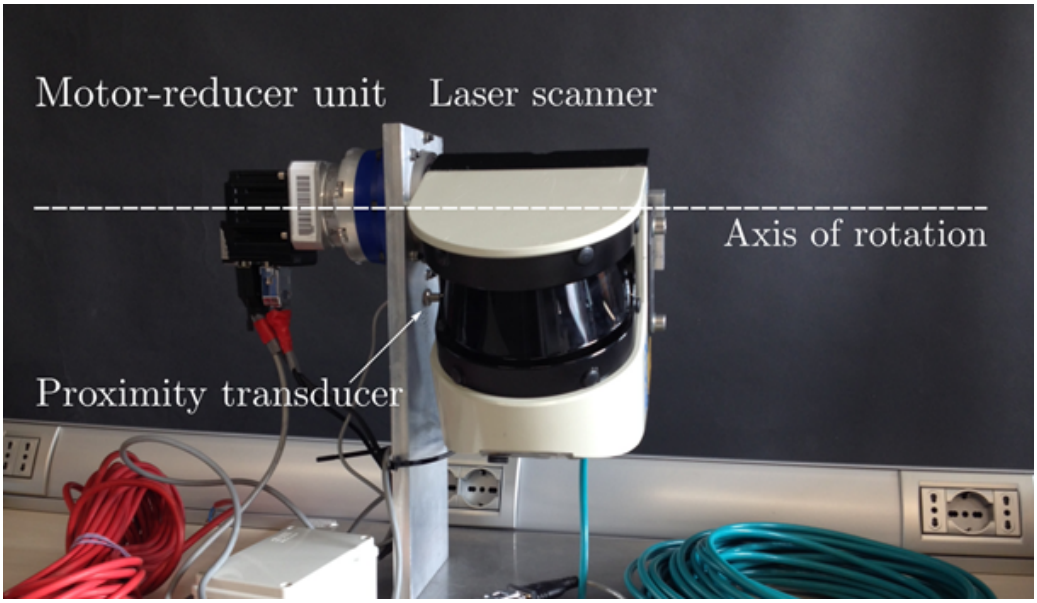


Figure 2.10: Final configuration of the SICK LMS 511 laser scanner

2.4 Description Of The Acquisition Unit

Acquisition and saving of the data was done through a dedicated software developed in LabVIEW environment, while the data processing was computed mostly through Matlab scripts. 1000 measurements were performed for each configuration tested, at a room temperature of around 20° C. A setup using a linear guide, as commonly exploited in [Ye and Borenstein, 2002], [Kneip et al., 2009] and [Alwan et al., 2005], could not be easily realized due to the extended measure range of interest (1 to 12 m). Two different setups were utilized depending on the type of test to be conducted.

2.4.1 Setup A

The experimental setup for tests regarding the influence of target distance and material properties is shown in Figure 2.11.

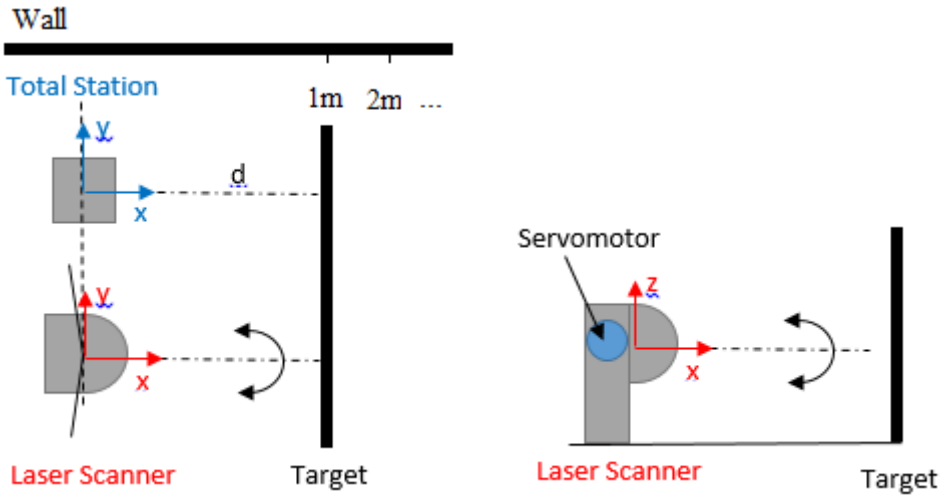


Figure 2.11: Setup A. Schematization of the setup components viewed from the top and from a side.

As rectilinear reference a wall of the wind tunnel of the Politecnico di Milano was considered, and using a Leica Total Station equispaced marks the ground were sketched. The laser scanner was placed parallel to the wall and the planar target perpendicular to it (see Figure 2.11 - left). In order to properly measure the distance to the target, a whole scan in the XY plane was registered and the beam presenting the minimum distance value was detected. Then, the tilt angle of the scanner was adjusted, moving the servomotor 0.5° each step up or down since a minimum value for the beam selected before was found (see Figure 2.11 - right). This procedure guaranteed that the scanner measured always the true distance, and that the target plane was always perpendicular to the wall while moving away from the scanner. The distances measured by the LMS 511 were compared to the ones obtained through the Total Station aligned to the scanner and whose accuracy (0.25 mm at 35 m) is by far smaller than the one of the laser scanner.

2.4.2 Setup B

A second setup was built up for the tests on the dependency of the angle of incidence between target surface and laser ray direction. Figure 2.12 shows it.

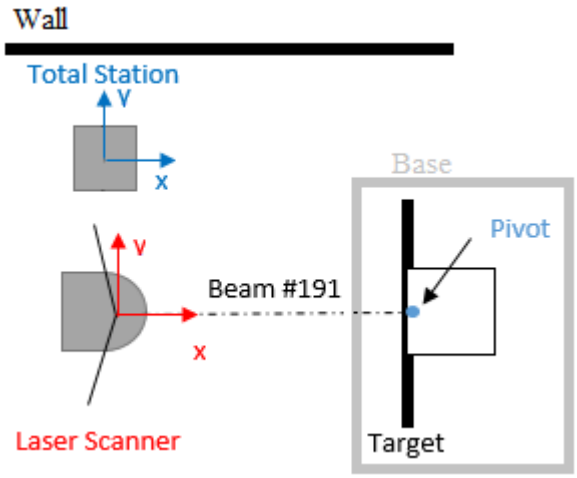


Figure 2.12: Setup B. Schematization of the setup viewed from the top.

This setup is composed of a planar base over which a second element can rotate on a point, ensuring that no translation is implied. This movement is obtained inserting a nail in the rotating element, that will be inserted in a corresponding hole in the base. Using well-shaped elements and good precision tools we can assure that the faces of the structure are parallel or perpendicular from each others. A metal plate is also added to tighten the grip of the nail on the rotating element. A goniometer is used to achieve a coarse measurement of the rotation, while the exact value is given by a best-fitting plane retrieved through a Singular Value Decomposition of nine sample points on the target scanned with the Total Station. During the orientation test the rotating element was changed to a board supporting the SICK LMS 511, while the target was held in the same place, allowing to check the response of different beams ensuring a target at the same distance.

2.4.3 Characterization of the LMS 511

This section presents the results of all the tests conducted. First, the effect of drift is analyzed; then the influence of the distance between target and sensor, the target properties (in terms of color and material), the ray orientation and the incidence angle are investigated. Considering the outdoor application, also the influence of lighting conditions is examined, and finally a test showing the mixed pixel problem is reported.

2.4.3.1 Drift Effect

Since the warm-up time is expected to increase with the increasing of the measured distance – as suggested by [Lee and Ehsani, 2008] -, the drift effect was analyzed placing the wooden target at the maximum distance of interest (12 m). The LMS 511 sampled 13500 full scans, over a stretch of time of about 2 hours. The results are visualized in Figure 2.13.

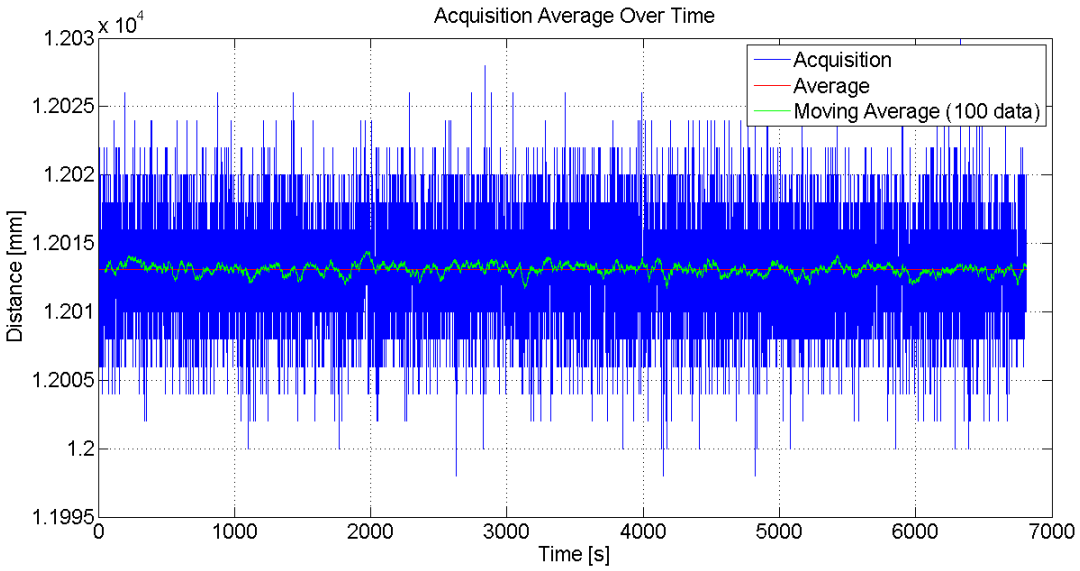


Figure 2.13: Drift Effect over 2 hours, there is no evident trend in the mean of the measures.

Note that there is not an evident drift effect, neither on the average value nor on the standard deviation of the measures (see Table 2.1). This means that the LMS 511 is a prompt sensor and does not need a warm-up time. This result is in contrast with the majority of the previously mentioned papers, where the authors highlighted an hour-minimum measurement drift.

The test was then repeated at a different distance to verify the results, and it confirms the above. Moreover, to reject the hypothesis of a warm-up time longer than two hours, an experimental campaign of 9 hours was conducted. Once again, the results were confirmed. Table 2.1 below presents the trend of the average distance for each hour and the correspondent standard deviation. No significant trend is evinced.

Hour #	Mean [mm]	Standard Deviation [mm]
1	2971.8	4.2
2	2971.2	4.0
3	2971.1	3.9
4	2971.0	3.9
5	2971.1	4.0
6	2971.2	4.0
7	2971.2	4.0
8	2971.2	3.9
9	2971.1	3.9

Table 2.1: Drift Effect on 9 Hours

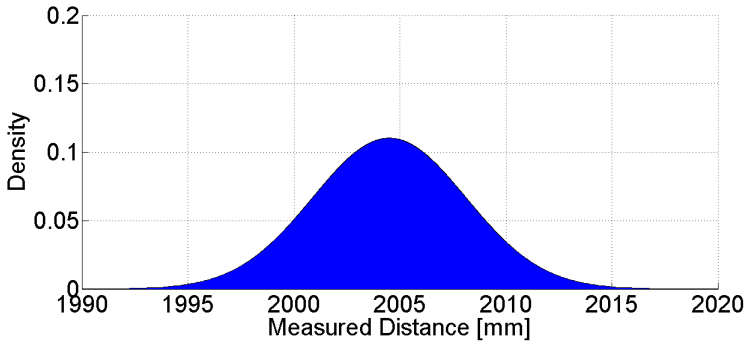


Figure 2.14: Data distribution from the drift effect test

Thanks to the high number of data registered in this test (more than 150000 acquisitions) it is possible to prove that the range values returned from the laser scanner follow a Gaussian distribution. Since that, in the next sections the results of tests can be compared in terms of their means and standard deviations.

2.4.3.2 Distance Effect

A planar target was placed in front of the scanner at different distances: from 2 m to 12 m with steps of 2 m each, following the procedure described in Section 2.4.1. Tests beyond 12 m were not performed as they would overcome the purpose of this paper. Figure 2.15 and 2.16 show the absolute and percentage errors and the standard deviations respectively.

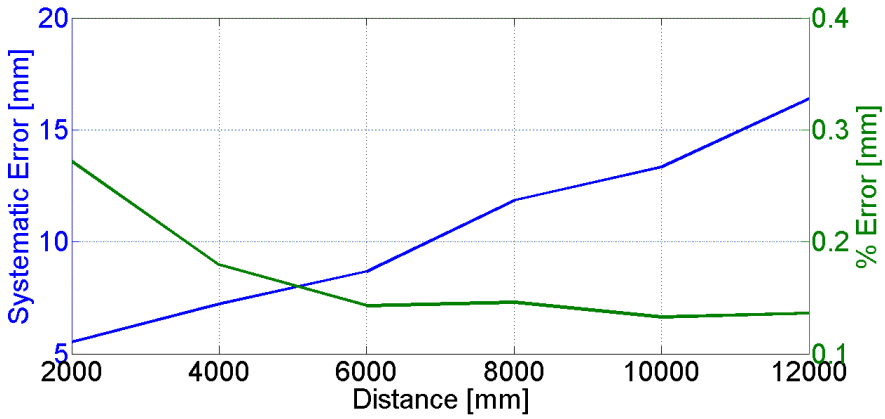


Figure 2.15: The Distance Absolute Error (blue), it increases almost linearly, while the Distance % Error in the range analyzed has an inverted proportionality relation with the distance.

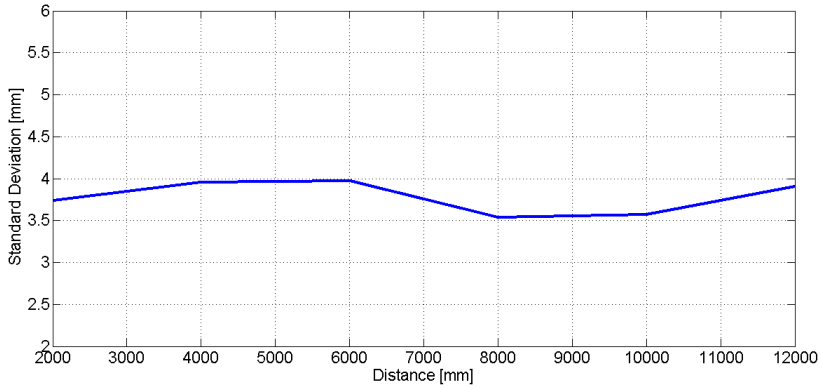


Figure 2.16: Distance Standard Variation, its trend does not vary significantly with the distance.

The standard deviation varies approximately from 3,55 mm to 3,95 mm. These values, that absolutely meet the manufacturer's specification, can be considered irrelevant compared to the distance measured.

2.4.3.3 Material Effect

Dependency on the target material properties is discussed in this section. The experimental setup is depicted in Section 2.4.1.

To allow the comparison of our results with the ones obtained by other researchers, we conducted experiments onto some common target materials:

- Wood
- Cardboard
- Aluminum
- Plastic

In addition, considering our specific application, other two targets were built up:

- Reflective tissue
- Sail tissue

Basically, for each material, the planar target was placed in front of the scanner at a distance of around 2 m.

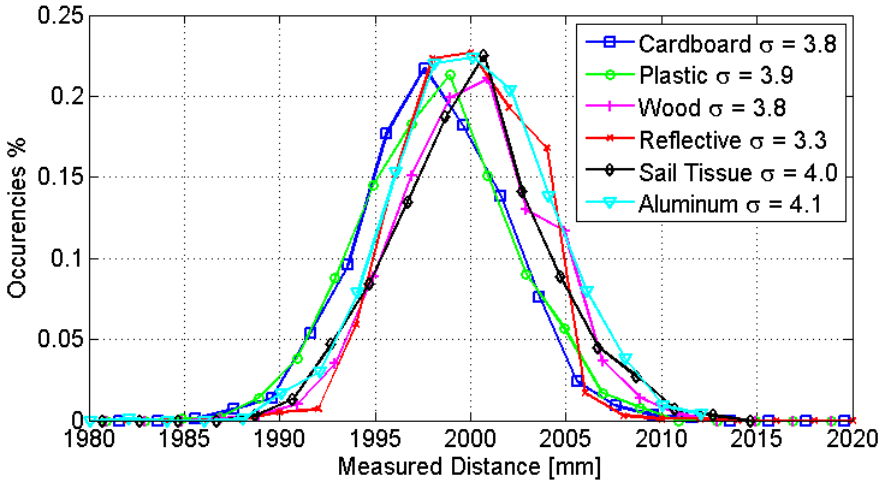


Figure 2.17: Distributions of ranges sampled for target covered in different materials.

The beam perpendicular to the target was identified by searching for the minimum measured distance among the ones corresponding to the target. 1000 whole scans were acquired for each material. Only distances registered by the perpendicular beam were considered. Figure 2.17 shows the distribution of the 1000 scans. Note that the material presenting the lowest standard deviation (3,3 mm) is the reflective one (red line). This could be due to the higher signal intensity that reaches the receiver. The worst material came out to be the sail tissue: standard deviation (4,1 mm, black line) . This is probably due to the transparency of the mold that did not reflect properly the laser signal. Figure 2.18 reports a picture of the sail highlighting the transparency of the surface (note the cloud beyond the sail).

Figure 2.18: Transparency of the Sail Tissue

2.4.3.4 Color Effect

In this section, the effect of the target color is discussed. Different colored adhesive films were pasted, one after the other, onto a glass planar surface placed at a fixed distance of 2000 mm far from the sensor. 1000 scans were acquired for each film. Once more, the data could be approximate by a normal distribution and mean values and standard deviations were computed. Various colored films were available, but we limited the analysis to the most interesting cases suggested by [Ye and Borenstein, 2002], [Kneip et al., 2009] and [Alwan et al., 2005]. In particular these cases are: green, yellow, black, and two different gray level films. Figure 2.19 shows the results for these tests.

Color	Mean [mm]	Standard Deviation [mm]	Nominal-Measured Distance [mm]
Green	1997.8	3.8	2.2
Yellow	2005.3	3.8	5.3
Black	2005.3	3.9	5.3
Light Gray	1997.7	3.7	2.3
Dark Gray	2014.7	4.0	14.7

Table 2.2: Values of the colors test comparison, the light gray film was the one with the narrowest response.

From these tests, we can state that the effect of the color does not influence significantly the measures. However, looking in detail, we can notice that:

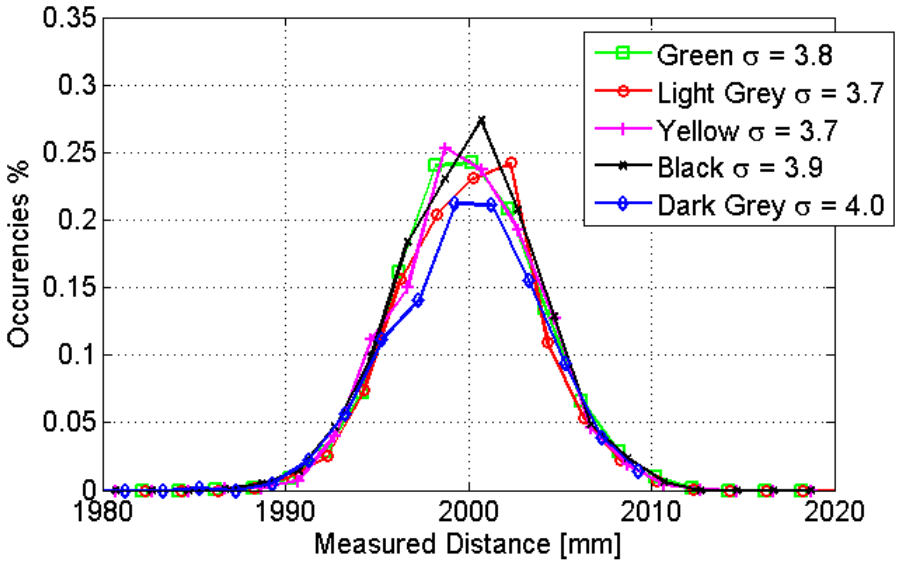


Figure 2.19: Distributions of ranges sampled for target placed at 2000 mm and covered with the adhesive film of different colors.

- dark gray and the black target present the highest values of spread (standard deviation respectively 4,0 and 3,9 mm)
- dark gray target presents the less precise measured distance (maximum displacement to the nominal distance – 14,7 mm-)
- light gray test presents the most precise measured distance (minimum displacement to the nominal distance – 2,3 mm-)

Thus, we can conclude that brighter targets lead to slightly better performances, probably because of a better reflection of the laser signal. The results obtained are in agreement with the ones reported in the previously mentioned works. Comparing the spread of the measures obtained testing the sail tissue (4,1 mm) to the one obtained in the light gray test (3,7 mm), we thought about coating the sail with this bright film to increase the quality of the measures on field.

2.4.3.5 Angle Effect

Considering Setup B (see Section 2.4.2), the wooden target placed at a distance of 1000 mm is rotated from 0° (target facing frontally the sensor) to 80°, with step of 10° each. The aim is to investigate the influence of the angle of incidence between the laser beam and the target surface. 1000 scans for each angle were acquired,

and only the distances retrieved by the central ray were considered. Afterward, the target was moved away from the scanner to reach the distances of 2000 mm and 3000 mm, and the procedure was repeated to support the first test. Figure 2.20 presents the standard deviation trends for the three distances as a function of the angle of incidence between the laser beam and the target surface.

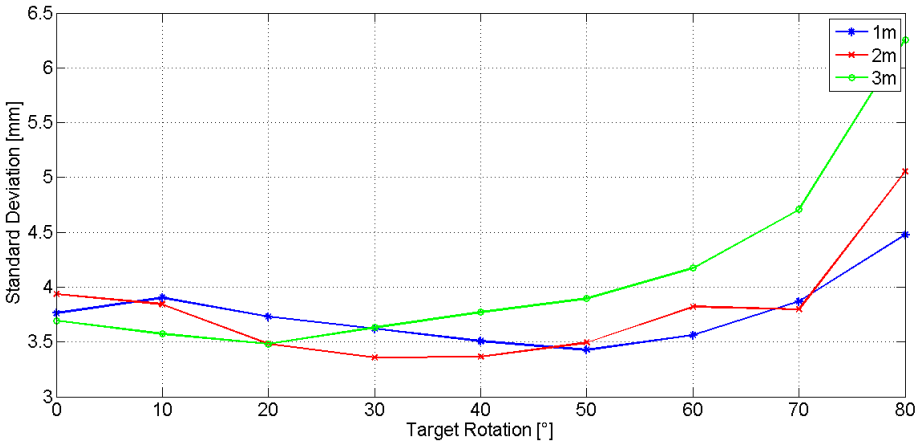


Figure 2.20: Angle Effect: influence of the rotation of the target on the standard deviation of the measures. Trends for tests at different distance are compared, in all tests over 70° the standard deviation values rise considerably.

Standard deviation trends look similar for the three tested distances and the values are comparable with the ones shown in the sections above, until an incidence angle of approximately 70°. A more oblique beam leads to a considerable increase in the data spread. This test was of particular interest to understand some faulty measures in the data acquired on the boat. In fact, the sensor placed astern and scanning the mainsail was not able to estimate a distance for some point on the top of the sail since the laser beams reached the surface with an angle beyond the 70°.

2.4.3.6 Angulation Effect

Differently from the previous test, it aims to clear the possible differences in data returning from different angle (i.e. different beams) to the scanner. Recalling the operating principle, these could be caused by imperfections in the rotating prism inside the scanner. For this test the scanner was placed onto the rotating base and the target was fixed at a distance of 2000 mm. The scanner was rotated and 1000 measurements were acquired for the beam perpendicular to the target. The setup is schematized in Figure 2.21.

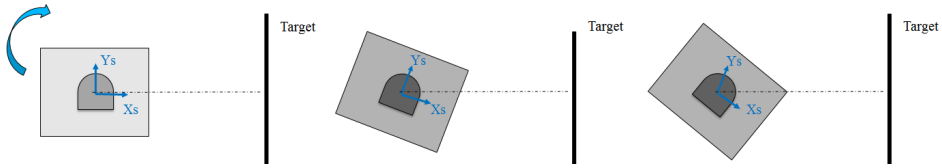


Figure 2.21: Angulation test setup: the scanner is progressively rotated

Results are showed in Figure 2.22, note that the standard deviation values resulting are all included between 3.7 mm and 4.05 mm, consistent with the previous considerations. On the other hand it seem that the sensor slightly overestimates the distance in the first section of the acquisition, from 0 to 60 degrees, and it underestimates it on the opposite side, from 90 to 180 degrees. Note that this effect could be due to a misalignment of the center of rotation used from the position of the reference axis of the SICK scanner. The manufacturer assured that they are placed exactly in the middle of the geometrical size of the unit, but the impossibility to open it prevented to verify this information.

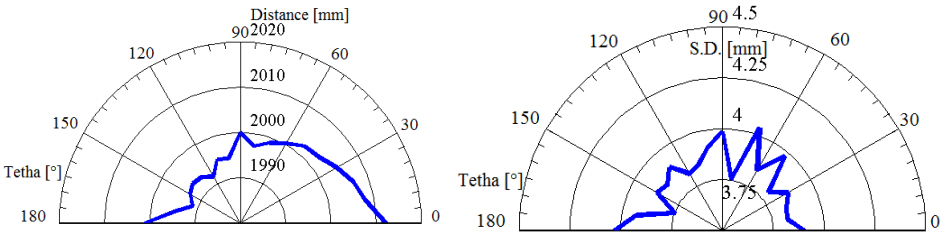


Figure 2.22: Effect of the angulation on the measure, the standard deviation is not in influenced, while the data acquisition seems to be affected.

2.4.3.7 Light Effect

Other faulty cases came out during the acquisitions on field because of the sun light that directly hit the scanner. Thus, the effect of an intense light source onto the sensor performances was investigated. To reproduce this situation a lamp was placed beyond the sail target, so that the source of light was completely covered by the sail tissue and then progressively moved close to an edge until it came out completely. A similar test was conducted outdoor, placing the laser scanner and the sail target in the sun light direction. As one can see from Figure 2.23, the light source impaired the measures. Moreover, as soon as the light entered the sensor,

the standard deviation increased almost three times.

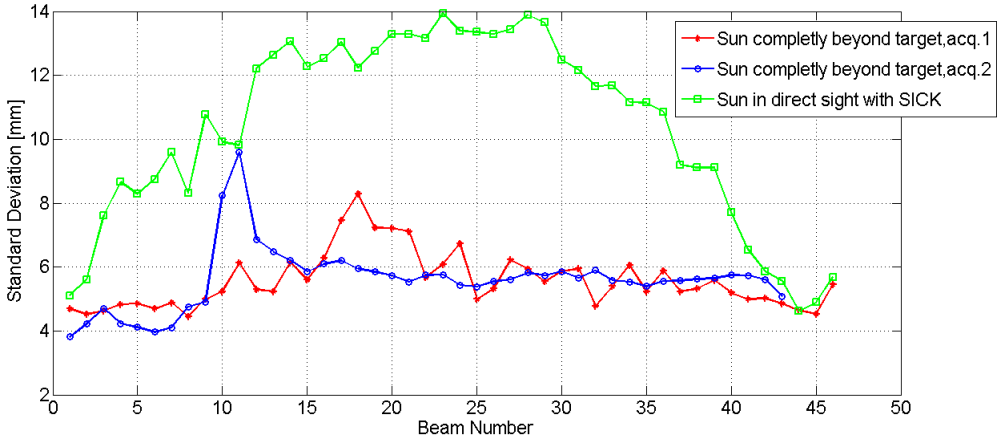


Figure 2.23: Light Effect: the peak in the blue and red measures corresponds to the position of the sun beyond the target. For every acquisition the standard deviation worsen with respect to a standard lighting case, but the worst case is when the sun hits directly the scanner (+ 350% Standard Deviation).

2.4.3.8 Mixed Pixel

The mixed pixel problem occurs when the spot mark of the laser beam, that hits the tested surface, falls on the edge of the target. For that point, the sensor averages the contribute of the signal reflected by the target and the one reflected by the background leading to an averaged estimated distance. Two tests were reported to verify that the mixed pixel problem occurs independently from the gap distance between target and background: test 1 considered a target placed at 1000 mm far from the sensor and a background at 2000 mm; test 2 considered a target placed closer to the background (at 1750 mm far from the sensor). Figure 2.24 shows these tests.

Both scans present non physical points in the gap between the planar target and the background due to the averaged distance performed by the scanner. Concerning our application, we can correlate the sail surface to the target and the sky to a background placed at an infinite distance from the sensor. Thus, we performed a test with a planar target oriented towards the sky. No mixed pixel error was evident. This is probably due to the fact that the only contribute that can be measured is the one coming from the spot mark onto the sail surface. This might be too low to let the sensor evaluate a distance and thus the point is lost.

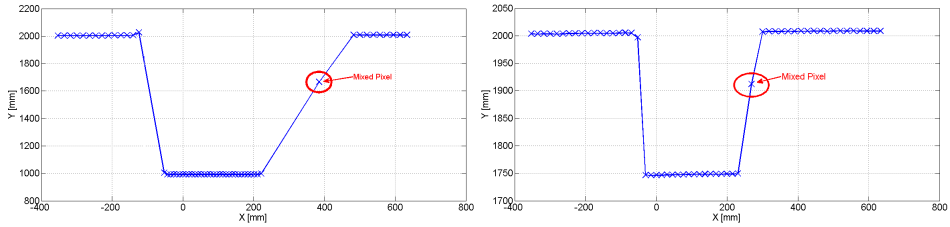


Figure 2.24: Mixed Pixel: two cases of occurrences. The target profile distortion is more prominent with the higher target-background distance.

2.4.4 Conclusions

This chapter aimed to displayed the main three-dimensional acquisition techniques currently available, characterize the measurement device, justifying the choice made for the measurement device, which ended up being the the LMS 511 Pro laser scanner manufactured by Sick. After the first acquisition campaigns the need for its characterization rose, due to some issue in data registration. Different tests to investigate several aspects that could affect its measures were performed. From the drift test it is concluded that the instrument does not suffer from a notable warm-up time, unlike others analyzed in previous literature. In the range considered (1m - 12m) the distance error is within the expected tolerance (max 0.27% of measure), while the standard deviation is bounded from 3.5 to 4 mm without an evident correlation with the distance. A comparison of the distribution of measures coming from different materials showed that the sail tissue does not have a good response with respect to others. As expected a reflective tissue has the lowest measures spread, and could be used to improve the field acquisition, wrapping the target with it. Another possibility is to use an adhesive film, as its color affect the quality of the measurement. From the different colors available, the best found are a light gray and yellow, generally leading to the conclusion that a brighter color leads to a better result. We showed that the effect of the incident angle rise with the increasing of the distance and that a limit angle, around 70°, is present, over which the accuracy of the laser scanner drops sensibly. The measure is more deeply influenced by the angulation from which the target has been recorded, as the distance is underestimated or overestimated in function of the acquisition angle. Moreover we proved that the lighting condition affects greatly the measures, as their standard deviation could increase by 50% where the sun is beyond a transparent target and up to 350% when the sun light hits directly the sensor. Finally same

cases where the laser scanner fails to correctly detect the target are displayed; these are due to the mixed pixel problem, which occurs when both the target and the background are within the scanner's range of field, while on outdoor test, without a fixed background this does not happen, but could lead to a loss of some data points. At the end of the characterization not only we have concluded that the considered scanner has met the manufacturer's specification but also found a quick and inexpensive way to improve the quality of the acquisitions.

Chapter 3

The Data Elaboration Procedure

The following chapter includes all the elaborations needed to transform a raw point cloud acquisition into an actual surface of a sail. Such transformations are:

- Rigid registration, which is the alignment of different scans into a common reference system;
- Scene interpretation, separating the samples representing the sail from the whole acquisition;
- Data re-sampling, re-sampling of the points help in removing the outliers and reducing measurement uncertainty
- Meshing, which is the creation of the surface starting from a set of points.

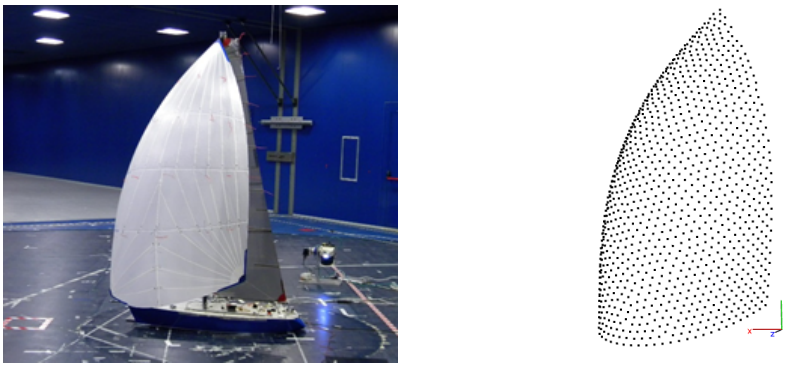
For each operation known algorithms are presented. Their performances are assessed first on synthetic data sets and on a sail model, and further on real acquisitions acquired both in wind tunnel and from on-field campaign. When a suitable algorithm for the purpose of the thesis is not found, a custom one is developed and tested. In the following, some basic concepts, that need to be clear for a better understanding of the work, are explained.

3.1 Basic Concepts

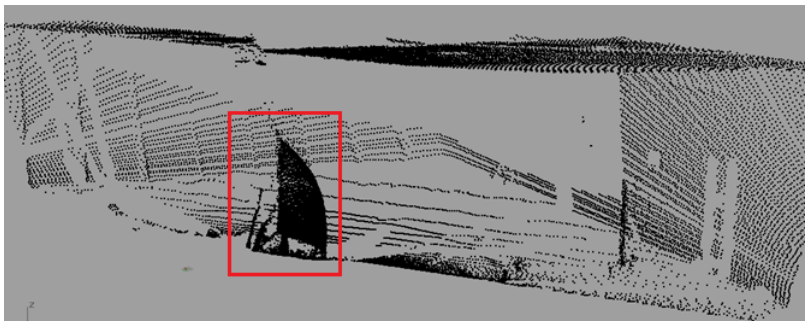
3.1.1 Point Cloud

A point cloud is a data structure used to represent a collection of multi-dimensional points. Commonly, a point cloud is a three-dimensional set that encloses the spatial

coordinates of an object sampled surface. Adding information about the color, the point cloud becomes 4D. Point clouds can be acquired using stereo cameras, time-of-flight scanners, depth cameras, or generated through computer programs synthetically. In Figure 3.1 two point clouds are presented: a model sail cloud (extracted from the CAD model), paired with a photo of its realization, and a raw point cloud coming from the acquisition device.



(a) Sail (left) and its point cloud model (right)



(b) Wind tunnel raw acquisition point cloud - the sail is highlighted in the red square

Figure 3.1: Point Cloud Examples

3.1.2 Neighborhood

The neighborhood of a point p is the set of its closest points, according to the Euclidean distance, and with a limit criterion. Points belonging to the neighborhood set are called neighbors. There are two types of neighborhood: if the limit is the distance threshold, it is *r-neighborhood* , which means that all the points inside a sphere of radius r are considered neighbors of p. The other way of creating a neighborhood is by number of neighbors. This means that only the k-th closest points to p are its neighbors; this is a *k-neighborhood*. The neighborhood is not an

object which is an end to itself, but it is used as initial step for retrieving further object information (see below).

3.1.3 Surface Normals

A surface normal, or simply normal, associated to a point p, is a vector perpendicular to the tangent plane that approximate the surface around p. The problem of

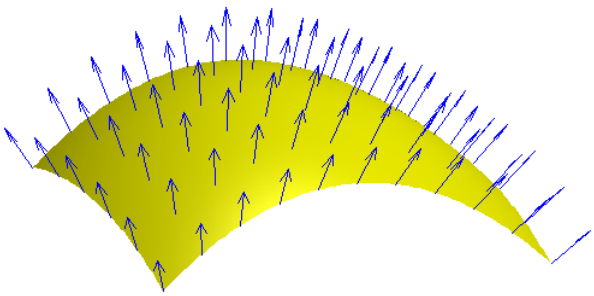


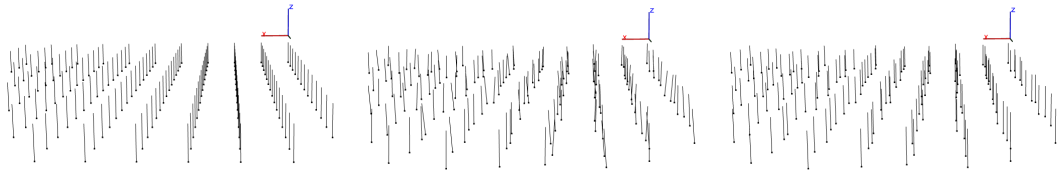
Figure 3.2: Surface Normals Example

determining the normal to a point on the surface is approximated by the problem of estimating the normal of a plane locally tangent to the surface, which in turn becomes a least-square plane fitting estimation problem. The solution for estimating the surface normal is therefore reduced to an analysis of the eigenvectors and eigenvalues (through the Principal Component Analysis, or PCA) of a covariance matrix created from the neighborhood of the query point. For each point p_i, the covariance matrix C is constructed as:

$$C = \frac{1}{k}\sum_{i=1}^{k}(p_i - \bar{p}) \cdot (p_i - \bar{p})^T,\ C \cdot \overrightarrow{v_j} = \lambda_j \cdot \overrightarrow{v_j},\ j \in \{0, 1, 2\} \tag{3.1}$$

Where k is the number of point neighbors considered in the neighborhood of p_i, $\bar{p}$ represents the 3D barycenter of the nearest neighbors, λ_j is the j-th eigenvalue of the covariance matrix, and $\overrightarrow{v_j}$ the j-th eigenvector. The normal $\overrightarrow{n_i}$ is the eigenvector corresponding to the smallest eigenvalue. $\overrightarrow{n_i}$ gives the direction of the normal, however there is no mathematical way to solve for the sign of the normal, so the orientation results ambiguous. This means that, for a same surface, some normals are oriented inwards and other outwards. A simple solution, proposed by Rusu in [Rusu, 2010], is to align all the normals orientations towards the origin of the axis. This is especially reasonable under the assumption that the point cloud comes from

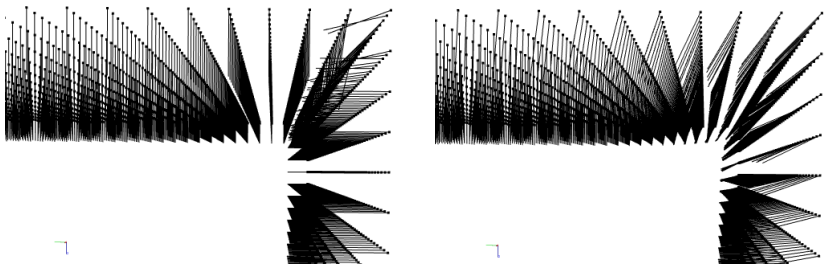
an acquisition device, which is placed in the origin of the axis. The choice of the neighborhood have a great influence in the output normals. The minimum number of neighbors of a point p needed to retrieve its normal is three, because a plane can be retrieved from a minimum of three samples. Using small neighborhood lead to the creation of normals which explain the local behavior of the surface, while a big neighborhood creates normals that represent the general trend of the surface. The size of the neighborhood has to be balanced in function of the point cloud considered. The presence of noise on the samples lead to imprecision in the normals value, if the neighborhood is too small, as visible from Figure 3.3. While a large neighborhood may help in reducing noise, on the other hand, it



(a) Plane With No Noise (b) Noisy Plane - k=5 (c) Noisy Plane - k=100

Figure 3.3: Plane Normals Example with k-neighborhood

may filter some important information about object local features, such as sharp edges or corners. They might be smoothed as shown in Figure 3.4 where normal directions change more gradually around the corner. More in detail, a synthetic point cloud representing a 90° angle between two planar surfaces has been generated and the normals were computed using two different neighborhood sizes (small k=5 and large k=100). Comparing the results, one can notice that in the second case



(a) k=5 (b) k=100

Figure 3.4: Edge Normals Example with k-neighborhood

normals do not follow the original surface, but tend to anticipate the 90° change.

The same results displayed in Figure 3.3 and Figure 3.4 can be obtained using a r-neighborhood, however the radius has to be adapted to the density of the point cloud, which has to be known in advance and has to be constant in the whole data set. These conditions are not always verified, so the use of the k-neighborhood is usually more robust. Normals are used in most of the processing methods as basic information, along with the spatial coordinates of the points. Whenever their use is requested, it should be reminded that they are not unique values, but depend on the neighborhood chosen, which then acts as an additional parameter to take into account.

3.1.4 Surface curvature

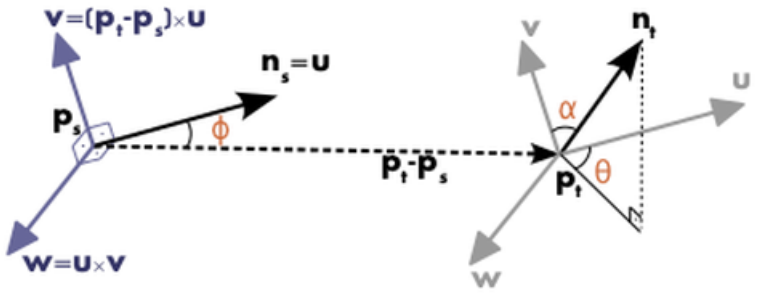
Intuitively, curvature is the amount by which a geometric object deviates from being flat. Rigorously, the curvature of a point p is a scalar number, found as:

$$\sigma = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{3.2}$$

Where λ_i are is the i-th eigenvalue of the covariance matrix created from the neighborhood of p, as explained in the previous section. The eigenvalues are ordered in ascending order, so λ_0 is always the smallest eigenvector.

3.1.5 Point Feature Histogram

Surface normal and curvature can provide local geometry information around a specific point. They are extremely fast and easy to compute, but sometimes not sufficient to identify object features. A Point Feature Histogram (PFH) representation, introduced by Rusu in [Rusu, 2010] is based on the relationships between the points in the k-neighborhood and their estimated surface normals. A PFH takes into account all the interactions between the directions of the estimated normals, defining a new local coordinate system (u,v,w), as explained in Figure 3.5, on the point query and expressing all the normals in the neighborhood as deviation from (u,v,w). The histogram describes the distribution of the deviation among the neighborhood. Using the local frame presented in Figure 3.5, the difference between the



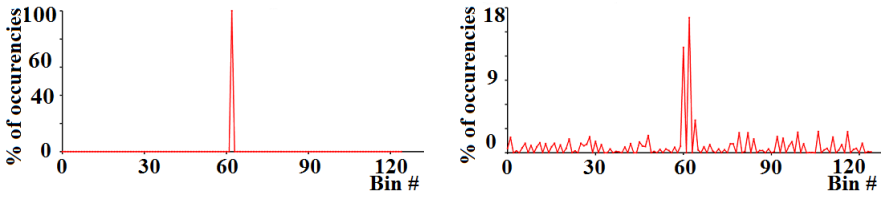
$$\begin{cases} u = n_s \\ v = u \times \frac{(p_t - p_s)}{\|p_t - p_s\|_2} \\ w = u \times v \end{cases} \tag{3.3}$$

Figure 3.5: Point Feature Histogram local coordinate system

two normals n_s and n_t can be expressed as a set of angular features as:

$$\begin{cases} \alpha = v \cdot n_t \\ \phi = u \cdot \frac{(p_t - p_s)}{d} \\ \theta = \arctan(w \cdot n_t, u \cdot n_t) \end{cases} \tag{3.4}$$

where d is the Euclidean distance between the two points p_s and p_t, $d = \|p_t - p_s\|_2$. The quadruplet $< \alpha, \phi, \theta, d >$ is computed for each pair of points in the neighborhood, reducing the 12 values (xyz and normal information, for each point) to four. For most cases, the fourth feature, d, does not present an extreme significance ([Rusu, 2010]), as for real acquisition d increases from the viewpoint and omitting d from the histograms has proved to be beneficial. For the creation of the actual histogram from the three features the reference is always Rusu's dissertation [Rusu, 2010]. Examples of the histogram are showed in Figure 3.6. The important property of PFH is that points that lie on similar surface have similar PFH. This means, for example, that points that lie on a plane have similar PFH. Points can then be classified as "point on a plane", "point on a sphere", "point near a corner", and so on. This information is much more accurate respect than the only normal information obviously.



(a) Plane PFH (b) Edge PFH

Figure 3.6: Point Feature Histograms Example

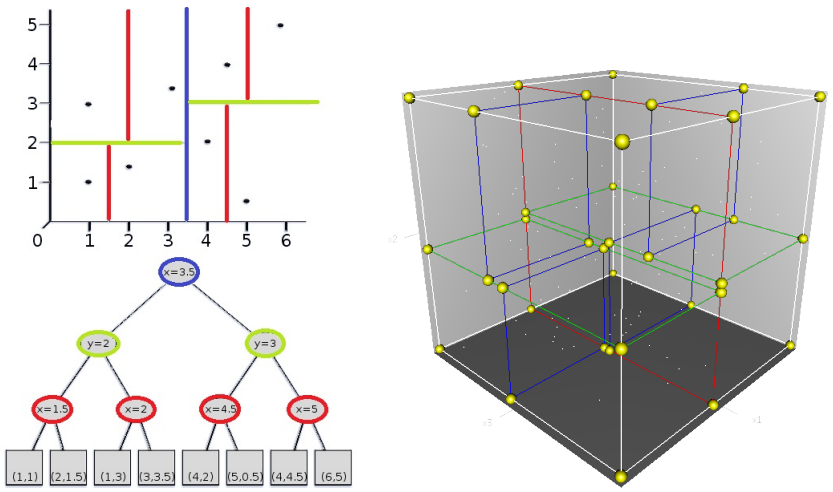
3.1.6 Fast Point Feature Histogram

The fast point feature histograms (FPFH) start from the PFH concept and reduce the computational complexity. The concept is similar to PFH but for more details refer to [Rusu, 2010] and [Rusu et al., 2009]. Main differencies from standard feature histograms are listed below:

- FPFH does not fully interconnect all neighbors of the considered point and thus some value pairs, which might contribute to capture the geometry around the query point, are missing.
- the PFH models a determined surface around the query point, while PFPH includes additional point pairs outside the neighborhood (the maximum distance is limited).
- the overall complexity of FPFH is greatly reduced, making possible to use it in real-time applications.
- the resulting histogram is simplified by decorrelating the values, that is simply creating separate feature histograms, one for each feature, and concatenating the together.

3.1.7 K-d tree

The K-d tree (k-dimensional tree) is a data structure used in computer science for organizing points in a space with k dimensions. It is a binary search tree with additional constraints imposed. The algorithm iteratively splits the space into two parts and builds a tree structure. Each point of the cloud is associated to a bounding box of the last level that encloses it, but thanks to the tree structure is linked to the all other larger boxes of the higher levels. K-d trees are, thus, very useful for range and nearest neighbor searches.



(a) 2D K-d tree visualization and corresponding tree structure

(b) 3d K-d tree visualization - the first division is in red, the second in green and then blue

Figure 3.7: K-d tree examples

3.1.8 Octree

An octree is a tree-based data structure for managing sparse 3-D data, with the same purpose of the K-d tree. The difference is that each node of the octree has exactly eight children, instead of two. Dividing the space in eight children means that the cubic bounding box of the data is split in eight equal cubes iteratively for each level.

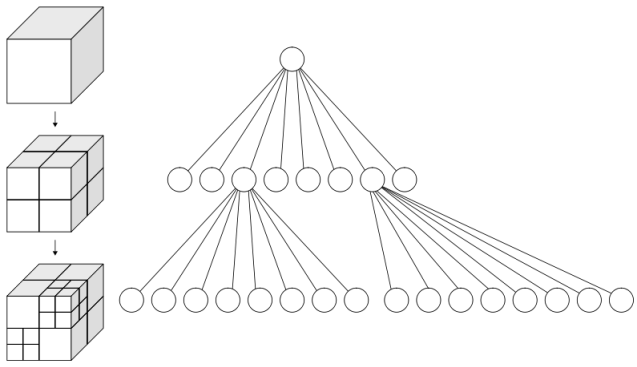


Figure 3.8: Octree Visualization and corresponding tree structure

3.2 Multiple Point Clouds Registration

The problem of consistently aligning various 3D point cloud data views into a complete model is known as *registration*. Its goal is to estimate the relative positions and orientations of the separately acquired views in a global coordinate framework, and to find a way to align and merge them together into a single point cloud so that the intersecting areas overlap perfectly. The work presented in this section is motivated by finding correct point-to-point correspondences in real-world noisy data scans, and estimating rigid transformations that can rotate and translate each individual scan into a consistent global coordinate framework. A motivation example in this sense is given in Figure 3.9, where two separate scans from two TOF laser scanners, one dedicated to the mainsail and the other to the offwind sail, are merged to form a unique point cloud representing the whole yacht model.

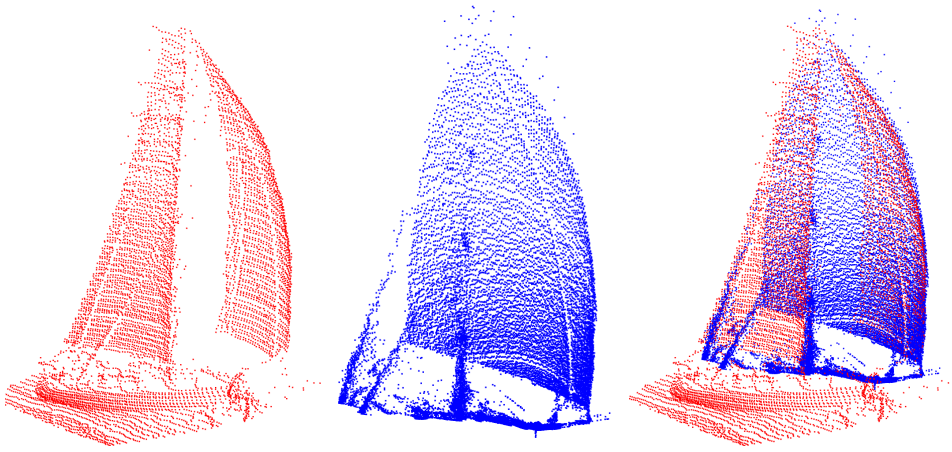


Figure 3.9: Example of registration: two point clouds representing the yacht model from different views (left and center) and the merged model (right)

The registration problem becomes easily solvable if point to point correspondences are perfectly known in the input data sets. This means that a selected list of points $p_i \in P_1$ have to "coincide" from a feature representation point of view with another list of points $q_j \in P_2$, where P_1 and P_2 represent two partial view data sets. Differently said, the sets of p_i and q_j have been sampled on the same real world surfaces, but from different acquisition poses. This means however, that in the complete point cloud model that needs to be created, they could be merged together, especially if their coordinates are equal $p_i = q_j$, and thus reduce the number of points overall. Because the quality of the data sets is influenced by sensor noise and other perturbing factors, the coordinates of the points will almost never be

equal unfortunately, and the above simplification will not hold. In the next sections the most common algorithms for point cloud registration are briefly introduced, and results onto artificial point clouds are presented. Considering the strengths of every algorithm, a registration procedure dedicated to the clouds acquired in the wind tunnel has been developed and described in the last Section.

3.2.1 SVD Based Registration Algorithm

One of the most popular registration methods for unorganized point cloud data sets is based on the singular value decomposition (SVD) of a matrix containing the three dimensional coordinates of corresponding points [Arun et al., 1987]. The singular value decomposition of a real or complex matrix $m \times n$ M is a factorization in the form:

$$M = U\Sigma V^T \tag{3.5}$$

where U is an $m \times m$ real or complex unitary matrix, Σ is an $m \times n$ rectangular diagonal matrix with non-negative real numbers on the diagonal, and $\mathrm{V^T}$ is an $n \times n$ real or complex unitary matrix. The diagonal entries σ_i of Σ are known as the singular values of M, from which the name of the method. When M is an $m \times m$ real square matrix with positive determinant, U, $\mathrm{V^T}$, and Σ are real $m \times m$ matrices as well, Σ is a scaling matrix, and U, $\mathrm{V^T}$ are rotation matrices. Imposing a unitary scale factor, U, $\mathrm{V^T}$ define the registration matrix ([Arun et al., 1987], [Soderkvist, 2014]), which can be expressed as a unique rotation and a translation, obtained as:

$$R = V \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & det\left(V \cdot U^T\right) \end{bmatrix} \cdot U^T \tag{3.6}$$

$$T = p_2 - R \cdot p_1 \tag{3.7}$$

where $\mathrm{p_2}$ and $\mathrm{p_1}$ are the center of mass, respectively, of the target data set and the source data set. The input matrix M, is composed from other two matrices A and B, through these steps:

$$\bar{\mu_x} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \bar{\mu_y} = \frac{1}{N}\sum_{i=1}^{N} y_i \qquad \bar{\mu_z} = \frac{1}{N}\sum_{i=1}^{N} z_i \tag{3.8}$$

$\bar{\mu_R} = [\bar{\mu_x}, \bar{\mu_y}, \bar{\mu_z}]$ is the barycenter of the reference point cloud and respectively $\bar{\mu_F}$ the barycenter of the registering cloud.

$$A = \begin{bmatrix} x_1^R - \bar{\mu_{Rx}} & y_1^R - \bar{\mu_{Ry}} & z_1^R - \bar{\mu_{Rz}} \\ .. & .. & .. \\ x_m^R - \bar{\mu_{Rx}} & y_m^R - \bar{\mu_{Ry}} & z_m^R - \bar{\mu_{Rz}} \end{bmatrix} \tag{3.9}$$

$$B = \begin{bmatrix} x_1^F - \bar{\mu_{Fx}} & y_1^F - \bar{\mu_{Fy}} & z_1^F - \bar{\mu_{Fz}} \\ .. & .. & .. \\ x_m^F - \bar{\mu_{Fx}} & y_m^F - \bar{\mu_{Fy}} & z_m^F - \bar{\mu_{Fz}} \end{bmatrix} \tag{3.10}$$

$$M = B * A^T \tag{3.11}$$

Since, the SVD procedure does not require all the cloud points, but matches only few of them (at least three pairs of points), it is not influenced by the number of points in the clouds, neither by their initial relative position. However the choice of the samples is critical: if not corresponding points are selected, a wrong registration occurs. Thus, the presence of noise in the data sets influences the algorithm results.

The relative position of the point picked also influence a good result of the algorithm, in fact, choosing points close to each others can lead to the calculation of singular matrices, which obviously unwanted. It is usually good practice to choose the samples as distanced as possible.

The SVD algorithm has been tested for the artificial scene described in Figure 3.10, where a rotation of 45° around one axis was imposed to the original cloud. Four matching pairs of points have been manually selected from the input data sets (highlighted in red in the figure), and the rigid registration has been computed. The resulting transformation matrix is reported in (3.12).

$$\begin{bmatrix} 0.707 & 0 & 0.707 & 0 \\ 0 & 1 & 0 & 0 \\ -0.707 & 0 & 0.707 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0.707 & 0 & 0.707 & -0.001 \\ 0 & 1 & 0 & 0.001 \\ -0.707 & 0 & 0.707 & -0.003 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

Table 3.1: Transformations matrix for noise free data sets - Imposed transformation

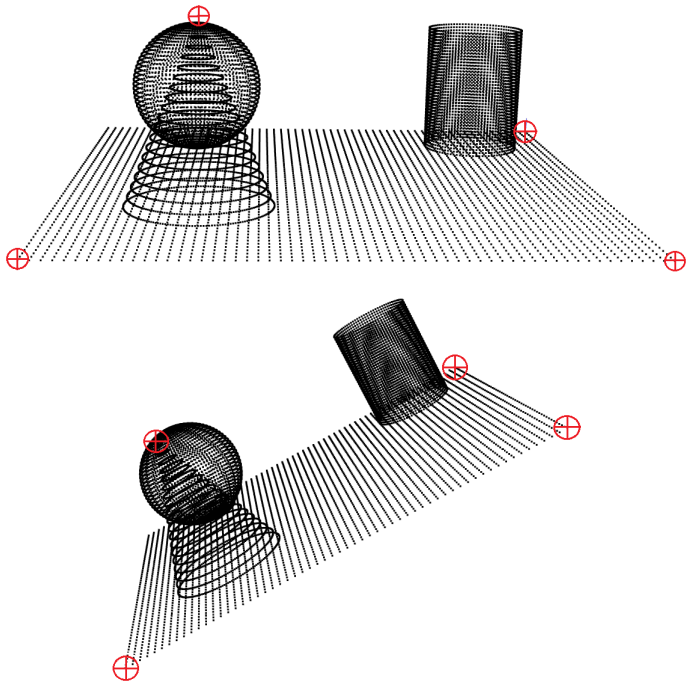


Figure 3.10: SVD scene: the point chosen for the registration are highlighted in the reference cloud (top) and registering cloud (bottom)

The transformation matrices correspond almost perfectly, aside from minimal errors, due to numerical operations performed by the machine. The noise effect is evaluated in the further test; the same scene presented in Figure 3.10 is considered, but noise is added to every point. The noise values follow a Gaussian distribution with mean equal to zero and standard deviation $\sigma = 0.05$ (a.u.). So that, the maximum level of noise (3σ) is equal to 0.5% of the distance between the origin of the reference system and the furthest point in the synthetic data set.

$$\begin{bmatrix} 0.707 & 0 & 0.707 & 0 \\ 0 & 1 & 0 & 0 \\ -0.707 & 0 & 0.707 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0.700 & 0.006 & 0.714 & 0.036 \\ -0.002 & 1 & 0.007 & 0.281 \\ -0.714 & -0.004 & 0.700 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

Table 3.2: Transformations matrix for noisy data sets - Imposed transformation

As expected, the SVD based procedure performs worse than the previous case since there is not perfect correspondence between the sample points in the two clouds (three plan corners and vertex of the cone) due to the noise. The relative pose is correctly estimated, as the rotation part of the matrix is right, however the

error on the translation is notable, 56% of the spatial resolution of the reconstructed cloud.

Another test was done in order to verify that the procedure is not influenced by the total number of points in the clouds to be registered. In this case the point cloud has been cut and rotated by 45 degrees, as visible in Figure 3.11.

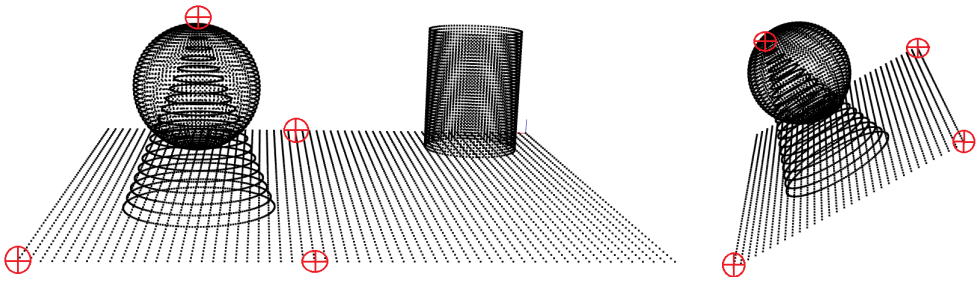


Figure 3.11: SVD scene for partial data sets: the point chosen for the registration are highlighted in the reference cloud (left) and registering cloud (right)

$$\begin{bmatrix} 0.707 & 0 & 0.707 & 0 \\ 0 & 1 & 0 & 0 \\ -0.707 & 0 & 0.707 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0.707 & 0 & 0.707 & 0.005 \\ 0 & 1 & 0 & -0.001 \\ -0.707 & 0 & 0.707 & 0.006 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

Table 3.3: Transformations matrices for poor overlapping data sets - Imposed transformation

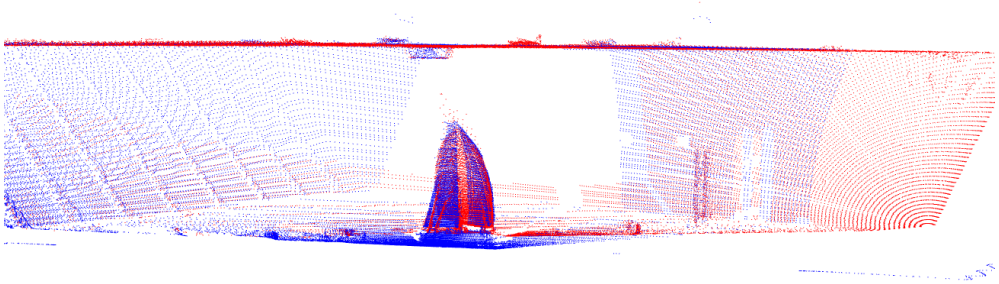
As expected, the presence of few points does not impair the registration whenever a few correspondence are perfectly known.

The test conducted on the real acquisition lead to the same conclusions, as notable in Figure 3.12a where the clouds can not be perfectly aligned due to the noisy scans.

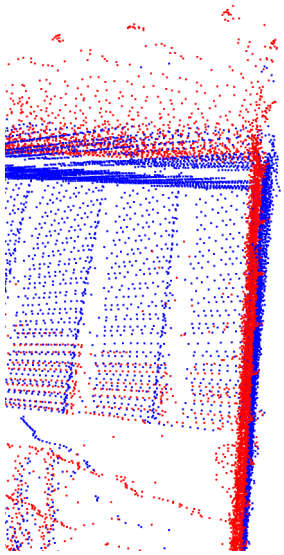
The faulty alignment is visible especially looking at the wind tunnel walls as shown in Figure 3.12b and in the gennaker luff, in Figure 3.12c.

$$\begin{bmatrix} 0.942 & -0.239 & 0.235 & -576.20 \\ 0.335 & 0.681 & -0.651 & -1886.35 \\ -0.004 & 0.692 & 0.722 & -192.41 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.15}$$

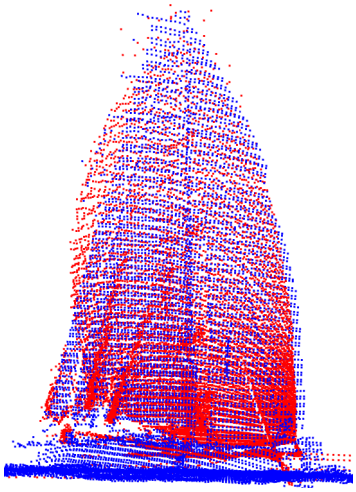
Table 3.4: Final transformation for real wind tunnel acquisition



(a) Final Registration



(b) Wall Close-Up, walls do not coincide



(c) Gennaker Side Close-Up, the right borders do not coincide

Figure 3.12: SVD Registration

Resulting distances between points belonging to a cloud with respect to the relative closest points belonging to the other clouds are more then ten times bigger than the acquisition accuracy. They can not be ignored and lead to discharge the SVD method for the real acquired cloud registration.

3.2.2 ICP Algorithm

Another widely exploited method for rigid registration is the Iterative Closest Point (ICP) algorithm [Besl and McKay, 1992], [Zhang, 1992], an iterative descend method which tries to find the optimal transformation (rotation matrix R and translation vector T) between two data sets $p_i \in$ P1 and $q_j \in P_2$ by minimizing the Euclidean distance error metric between their overlapping areas.

$$min \sum_{i=1}^{n} \|R \cdot p_i + T - q_i\|^2 \tag{3.16}$$

Modified algorithms, considering not only 3D spatial information, have been implemented: [Johnson and Kang, 1999] takes into account the colors acquired as well, [Eggert et al., 1998] performs the simultaneous registration of multiple range views, [Declerck et al., 1997] carries out 3D-2D projective transformations. [Ristic and Brujic, 1997] instead proposes to use a triangular mesh model approximation to accelerate the algorithm and suggests to assume the standard uncertainty of the device used to perform the measurement as tolerance to terminate the ICP iterations.

Whichever the approach considered, the ICP performs the steps in Algorithm 3.1.

Algorithm 3.1 ICP Algorithm

1. The set Q of closest points between the source data set S_d and the target dataset T_d is identified
2. The transformation matrices (R and T), minimizing the chosen cost function over Q, are determined.
3. The transformation is applied to the measured points set as $S_d' = R * S_d + T$.
4. If change in the cost function is greater than a preset value the procedure is repeated from the point 1.
5. Else the procedure ends.

[Besl and McKay, 1992], proved that ICP algorithm converges always to a local minimum, it means that the algorithm is able to provide the optimal solution only if the relative position of the cloud data set is close. Clouds that start the procedure presenting very different spatial positioning are not likely to be correctly registered, while on the other hand, ICP can achieve fine accuracy, when there is only a little displacement between the two data sets. As a good initial relative position can be consider the case in which the cloud centers of mass are close to each other and the relative cloud orientation is small. In order to better understand the reason beyond this, step (2) has to be further investigated. For each iteration, the procedure estimate the best rotation matrix R, performing a singular value decomposition of

the matrix M = B*A' (see Section 3.2.1 for details) containing data belonging to a cloud A and the relative closest points in cloud B. [Besl and McKay, 1992] also proved that the score function can be expressed as function of the cloud centers of mass, μ_{corr} and μ_{reg} and that the optimal translation vector applied to the registering cloud, at every iteration can be written as:

$$\overrightarrow{q_t} = \mu_{corr} - R \cdot \mu_{reg} \tag{3.17}$$

The construction of the correspondence set B is the critical step: for each point of the cloud to register the closest point (in terms of Euclidean distance) of the model cloud is assigned as "correspondent". If the clouds are poorly aligned, a model point could be matched to several points of the registering data set. This leads to the creation of a correspondence set composed of several identical points, which, in turn, leads to incorrect transformation matrices. For every iteration step the registering cloud is first rotated and then its barycenter is superimposed on the correspondence set barycenter, as (3.17). If the correspondences set is not correctly built (i.e. bad initial position) the translation moves the registering cloud not toward the model center of mass. This behavior in not bad in itself, as it still moves the cloud closer the model but the subsequent ICP iteration might reach a local minimum ending the procedure prematurely. In other words, this means that the algorithm won't move anymore the data because any movement would lead to an increasing of the cost function, although globally another minimum (i.e. better relative position) exists.

This issue is well known and discussed in many works, as [Chen and Medioni, 1991], [Park and Subbarao, 2003] and [Pottmann et al., 2004]. In the latest Pottmann et al. analyzed the problem and proposed a procedure that takes into account also information about the point normals and obtains the transformation matrices through an instantaneous kinematics approach rather than SVD. The use of instantaneous kinematics, however, only simplified the processing operation of the procedure proposed by Chen and Medioni in [Chen and Medioni, 1991]. They apply a motion to the cloud such that the sum of the squared distances from the reference cloud is minimal. For each point x, its velocity vector is defined as:

$$\mathbf{v}(x) = \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x} \tag{3.18}$$

where $\bar{\mathbf{c}}$ is the velocity vector of the origin of the reference coordinate system

and $\mathbf{c}$ represents the vector of angular velocity (or Darboux vector). Up to the first differentiation order, every three-dimensional movement can be expressed locally as translation with constant velocity ($\bar{\mathbf{c}} = 0$), a uniform rotation around an axis ($\mathbf{c} \cdot \bar{\mathbf{c}} = 0$) or an uniform helical motion ($\mathbf{c} \cdot \bar{\mathbf{c}} \neq 0$). A generic velocity vector field is hypothesized, such that the corresponding velocity vectors $\mathbf{v_i(x)}$ for each point minimize the quadratic cost function F:

$$min \sum_{i=1}^{N} F_i(\mathbf{x_i} - \mathbf{v}(\mathbf{x_i})) \tag{3.19}$$

From the velocity vector field a transformation that displaces the points $\mathbf{x}_\mathrm{i}$ in the same way as the velocity vector would do is calculated, through a linearization of the motion. The set of correspondences is required as before, but also the normals $\mathbf{n_i}$ of the corresponding points are calculated and stored. The motion is considered as general as possible, so it would be a helical motion in the form (3.18). The generic distance from the point $\mathbf{x}_\mathrm{i}$ to its matching point $\mathbf{y}_\mathrm{i}$ can be expressed as function of their initial distance d_i and the velocity vector as:

$$d_i + \mathbf{n}_i \cdot (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i) \tag{3.20}$$

the cost function (3.19) can be written as

$$min \sum_{i=1}^{N} (d_i + \mathbf{n}_i \cdot (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i))^2 \tag{3.21}$$

The minimization can be solved using a system of linear equations, rewriting (3.20) as:

$$d_i + \mathbf{n}_i \cdot \bar{\mathbf{c}} + (\mathbf{x}_i \times \mathbf{n}_i) \cdot \mathbf{c} = d_i + (\mathbf{x}_i \times \mathbf{n}_i, \mathbf{n}_i) \begin{pmatrix} \mathbf{c} \\ \bar{\mathbf{c}} \end{pmatrix} = d_i + A_i C \tag{3.22}$$

and (3.21) as:

$$F(C) = min \sum_{i=1}^{N} (d_i + A_i C)^2 = \sum_i d_i^2 + 2 \sum_i d_i A_i C + \sum_i C^T A_i^T A_i C \tag{3.23}$$

$$= D + 2B \cdot C + C^T AC$$

The unique minimum of the quadratic cost function is a column vector C, with six entries (since A_i is, by construction, a general positive six-by-six matrix) which solves the linear system

$$AC + B = 0 \tag{3.24}$$

where $A = A_i^T A_i$ and $B = 2 \sum d_i A_i$ can be obtained from the registering points and their corresponding point normals. The rotation and translation matrices can be recovered from the elements of C, with the relations (3.25) and (3.26), since at the start of the procedure, an helical motion was supposed. For more details on helical motion refer to [Pottmann and Wallner, 2009].

$$R = \begin{bmatrix} c2 * c3 & c3 * s1 * s2 - c1 * s3 & c1 * c3 * s2 + s1 * s3 \\ c2 * s3 & c1 * c3 + s1 * s2 * s3 & c1 * s2 * s3 - c3 * s1 \\ -s1 & c1 * s1 & c1 * c2 \end{bmatrix} \tag{3.25}$$

$$\text{where} \quad \begin{matrix} c1 = \cos(C(1)) & c2 = \cos(C(2)) & c3 = \cos(C(3)) \\ s1 = \sin(C(1)) & s2 = \sin(C(2)) & s3 = \sin(C(3)) \end{matrix}$$

$$T = \begin{bmatrix} C(4) \\ C(5) \\ C(6) \end{bmatrix} \tag{3.26}$$

Visually speaking, in the first ICP algorithm presented ([Besl and McKay, 1992]) the cost function minimizes the distance from points in the cloud to register and their correspondences, while the second one minimizes the distance between the point and the target surface (i.e. the distance from the plane tangent to the target surface in the correspondence point). This is why the first is usually referred as point-to-point ICP, while the other as point-to-plane ICP. Both of them rely on the construction of the correspondence set, which is a computational expensive step, since has to find a correspondence for each point of the cloud to register, comparing it to every point of the reference cloud. In [Rusu, 2010], Rusu proposed an alternative construction of the correspondences set, through the use of Point Feature Histograms (see section 3.1.5 for details). Briefly, PFH are histograms of the dis-

tribution of the normals in a point neighborhood; they can identify peculiar points, like corner points, edge points, points lying on a plane or on a curved surface, etc. His assumption is that corresponding points from different data set should have at least similar PFH (feature persistence hypothesis) and that it is not necessary to compare the whole data sets, but only the points whose PFH are less common. For example, for registering two cones, it should be sufficient to match the vertices and the lowest circumference edge points. In this way, the computational cost of matching points significantly drops, since number of the points compared are usually at least cut in half, respect to a classical ICP method. In his implementation, used in the PCL libraries, a registering point $\mathbf{p}_\text{i}$ is matched with another one $\mathbf{q}_\text{i}$ from the target cloud, selecting it randomly from the list of points $\mathbf{Q}$ whose PFH are similar to $\mathbf{p}_\text{i}$ PFH. Once the points has been matched, a point-to-plane registration is performed, using the cost function:

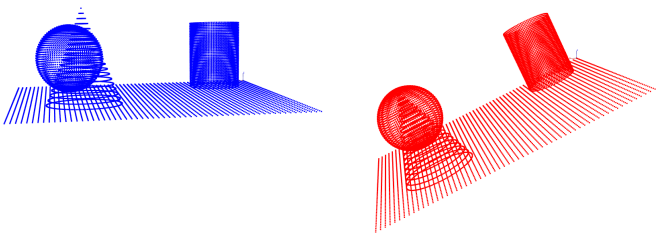
$$min \sum_{i=1}^{N} \|(R \cdot \mathbf{p}_i + T - \mathbf{q}_i) \cdot \mathbf{n}_{\mathbf{q}_i}\| \tag{3.27}$$

which is comparable to (3.21). The random selection of the correspondences might seem suboptimal, but it is a decision made to speed up the computational time. This algorithm then proceeds to complete the registration with the same steps explained for the point-to-plane ICP. Both [Besl and McKay, 1992] and [Zhang, 1992] found that the ICP algorithm has fast convergence (i.e. large transformation) in the first iterations but it is strongly dependent from the initial position. Thus, it is often referred as a *finely registration algorithm*, which comes after a first rough alignment. All the algorithms perform an iteration until an end condition is verified; most of the time this condition is represented by a maximum number of iterations allowed or a minimum cost function value to reach that estimates the convergence in the procedure.

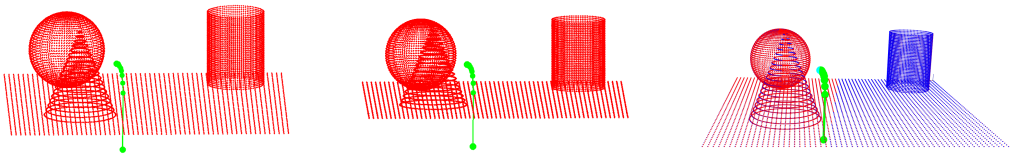
Tests were conducted on point cloud representing the same scene with regular geometry objects as in the previous section, as visible in Figure 3.13a. Then, a rotation of 30° around the Y axis has been imposed to the original point cloud, as visible in Figure 3.13b, and the registration via different ICP algorithms has been performed, retrieving the final transformation matrix. This matrix can be seen as the composition of a translation (represented by the contribute of the fourth column, row one to three), and a rotation, given by the 3×3 up left sub matrix.

The tests reported below compare the performances of different ICP algorithms,

both in terms of alignment result and time consumption. The values shown are retrieved from tests performed on the same computer, which has Intel Core 2 CPU 6600 @ 2.40GHz working on a 32-bit Linux operative system.



(a) Cloud 1 (b) Cloud 2



(c) ICP point-to-point registration - only Cloud 2 is visible due to the perfect alignment

(d) ICP point-to-plane registration - only Cloud 2 is visible due to the perfect alignment

(e) PFH ICP registration - Clouds very close to each other

Figure 3.13: Artificial point clouds before (up) and after (down) registration, the green curve represent the path taken by the cloud baricenter through the iteration steps

As shown in Figure 3.13c and (3.28), all the ICP procedures provide the correct alignment; the green lines and dots highlight the path and the position covered by the center of the cloud to register during the registration procedure at each iteration. Moreover, all the algorithm were tested setting as end iterative condition a fixed number of iteration (25 in this case) or a relative mean error between two consecutive iterations (set at 10^{-3}). The mean error is the squared sum of the distance for each point from its correspondence divided by the number of points considered. When the difference between the mean error in two consecutive iterations is smaller than a threshold it means that the algorithm is near a local minimum and convergence is reached, further iterations would not improve the resulting output transformation.

The performances are compared with two different tests, in the first, whose results are showed in Table 3.6, all the algorithms are ended after 25 iterations. In the second test, the ending condition is chosen as the relative mean error between

	Point	Plane	PFH
Iterations	25	25	25
Time [s]	150	226	7
Relative Error	10^{-16}	10^{-18}	10^{-9}

Table 3.5: ICP algorithms comparison for scenes described in Figure 3.13 - Iteration Number Ending Condition

	Point	Plane	PFH
Iterations	15	7	13
Time [s]	90	66	4.2
Relative Error	10^{-5}	10^{-5}	10^{-4}

Table 3.6: ICP algorithms comparison for scenes described in Figure 3.13 - Relative Error Ending Condition

two iterations. The relative error threshold is fixed to 10^{-3} after experimental tests. From Table 3.5 is evident the greater speed of the Point Feature Histograms ICP. The incredibly high accuracy of point-to-point and point-to-plane are due to the fact that for the test the same point cloud, rotated, is used both as registering cloud and target cloud, but this could not be expected for noisy data sets. Even though the PFH ICP have a smaller accuracy it retrieves successfully the correct transformation, as visible from the matrices comparison in Table 3.8. This results are expected from the theory previously explained. Since each procedure converges before reaching 25 iterations the matrices obtained in both the ending cases for a single ICP are the same; this proves that iterating after the convergence is useless. The trade-off between time and result suggest the PFH ICP as the best choice.

$$\begin{bmatrix} \cos(30) & 0 & \sin(30) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(30) & 0 & \cos(30) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.866 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \\ -0.5 & 0 & 0.866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.28}$$

Table 3.7: Transformations matrix for noise free data sets - Imposed rotation

$$\begin{bmatrix} 0.866 & 0.007 & 0.499 & -0.1 \\ -0.009 & 0.999 & -0.003 & 0 \\ -0.499 & 0.007 & 0.866 & 0.6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(a) Point-to-point ICP

$$\begin{bmatrix} 0.866 & -0.010 & 0.499 & 1 \\ 0.001 & 0.999 & -0.017 & 0.1 \\ -0.499 & 0.014 & 0.866 & 1.6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(b) Point-to-plane ICP

$$\begin{bmatrix} 0.866 & 0.007 & 0.499 & -0.1 \\ -0.009 & 0.999 & -0.002 & 0 \\ -0.499 & 0.007 & 0.866 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(c) PFH ICP

Table 3.8: ICP matrices retrieved for noise-free data sets, rotated one by each other

	Point	Plane	PFH
Iterations	25	25	25
Time [s]	174	258	11
Relative Error	0.027	10^{-9}	0.231

Table 3.9: ICP algorithms comparison for scenes described in Figure 3.14e - Iteration Number Ending

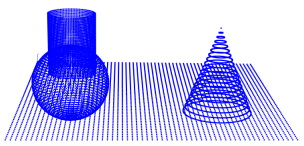
	Point	Plane	PFH
Iterations	37	16	44
Time [s]	222	143	19.3
Relative Error	10^{-4}	10^{-4}	10^{-4}

Table 3.10: ICP algorithms comparison for scenes described in Figure 3.14e - Relative Error Ending

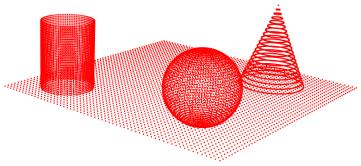
$$\begin{bmatrix} \cos(120) & -\sin(120) & 0) & 30 \\ \sin(120) & \cos(120) & 0 & 20 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -0.5 & -0.866 & 0 & 30 \\ 0.866 & -0.5 & 0 & 20 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.29}$$

Table 3.11: Transformations matrix for noise free data sets - Imposed transformation

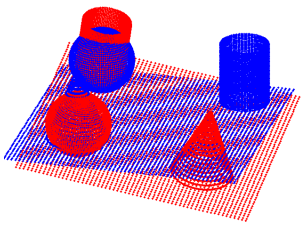
The sensitivity to the starting relative position of the two confronted data sets is the main flaw of the algorithm, to test its behavior a transformation composed by a rotation of 120° on the Z axis, followed by a translation on the X and Y



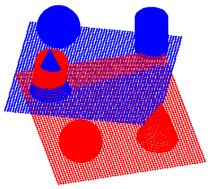
(a) Cloud 1



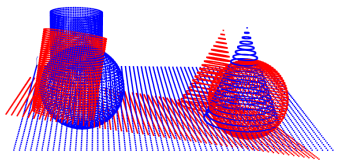
(b) Cloud 2



(c) Cloud 2 registered against Cloud 1 - Point to point ICP



(d) Cloud 2 registered against Cloud 1 - Point to plane ICP



(e) Cloud 2 registered against Cloud 1 - PFH ICP

Figure 3.14: Artificial point clouds rotated and translated before registering (up) and after (down)

axis of 30 and 20 (a.u.) respectively has been imposed to the same cloud used in Figure 3.13a. The translation applied has the purpose to keep the barycenter close, in order to highlight the influence of the different orientation of the scenes. The result of ICP in this case are displayed in Figure 3.14e and (3.12), both the image and the matrices comparison show that the algorithms do not find an appropriate solution. The matrices retrieved showed in Table 3.12 are the ones found with the convergence criterion ending. The issue of the initial position for ICP algorithms is hard to define and it is still discussed in the scientific community, especially because all the solution adopted lacks generality and have to rely on pre-knowledges on the composition of the scenes (presence of planes, guess on the relative position of the reference systems, etc.). Tables 3.9 and 3.10 shows that all algorithms manage to converge, even though with a higher number of iterations respect to the previous case.

$$\begin{bmatrix} 0.866 & 0.441 & -0.236 & -2.9 \\ -0.421 & 0.897 & 0.130 & -6.3 \\ 0.269 & -0.013 & 0.963 & -0.4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(a) Point-to-point ICP

$$\begin{bmatrix} 0.998 & 0.078 & -0.129 & -2.2 \\ -0.050 & 0.982 & 0.181 & 1.1 \\ 0.141 & -0.172 & 0.975 & 0.84 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(b) Point-to-plane ICP

$$\begin{bmatrix} 0.858 & 0.453 & -0.240 & -2.8 \\ -0.433 & 0.891 & 0.132 & -6 \\ 0.274 & -0.009 & 0.962 & -0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(c) PFH ICP

Table 3.12: ICP matrices retrieved for noise-free data sets, rotated and translated one by each other

$$\begin{bmatrix} \cos(30) & 0 & \sin(30) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(30) & 0 & \cos(30) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.866 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \\ -0.5 & 0 & 0.866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.30}$$

Table 3.13: Transformations matrix for noisy data sets - Imposed transformation

$$\begin{bmatrix} 0.866 & 0.006 & 0.500 & -0.1 \\ -0.009 & 1 & -0.004 & 0 \\ -0.500 & 0.008 & 0.866 & -0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(a) Point-to-point ICP

$$\begin{bmatrix} 0.864 & 0.074 & 0.499 & 0.7 \\ 0.008 & 0.999 & 0.029 & 0 \\ -0.499 & -0.029 & 0.866 & 1.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(b) Point-to-plane ICP

$$\begin{bmatrix} 0.866 & 0.006 & 0.500 & 0.1 \\ -0.009 & 0.999 & -0.004 & 0 \\ -0.500 & 0.008 & 0.866 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(c) PFH ICP

Table 3.14: ICP matrices retrieved for noisy data sets, rotated one by each other

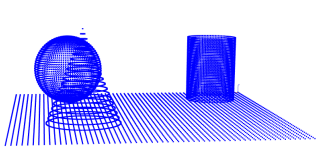
	Point	Plane	PFH
Iterations	25	25	25
Time [s]	166	247	8
Relative Error	10^{-7}	0	10^{-9}

Table 3.15: ICP algorithms comparison for noisy scenes, similar to Figure 3.14e - Iteration Number Ending

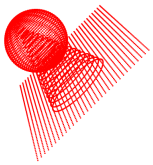
	Point	Plane	PFH
Iterations	16	6	15
Time [s]	107	64	6
Relative Error	10^{-4}	10^{-4}	10^{-4}

Table 3.16: ICP algorithms comparison for noisy scenes, similar to Figure 3.14e - Relative Error Ending

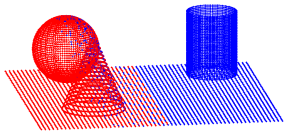
Also in this erroneous case the PFH ICP can be selected as best choice. The presence of a limited noise, uniformly added to the points coordinates and simulating the TOF scanner measure uncertainty for a scene acquired at 1 m, does not influence significantly the behavior of the algorithm, which can still recover the correct transformation, as visible from the comparison between (3.30) and (3.14). To achieve these results, the same clouds used for the first test are used, with the addition of a level of noise comparable to the values found in section 2.4.3. The value of the relative error for the point-to-plane ICP at the twenty-fifth iteration, exactly equal to 0, means that the procedure has perfectly converged (i.e. no further transformations), not that the registering points are perfectly matched. Moreover, ICP method assumes that each points in the reference cloud has a corresponding one in the other cloud. Failures, in facts, come out whenever the overlapping area between the two point clouds to be registered is too small with respect to the whole scene acquired in each scan or when a cloud is by far less populated than the other. In these cases, if the algorithm can not find a correspondence for each point and the procedure performs a wrong transformation even after hundreds of iterations, as presented in (3.31) and in Figure 3.15.



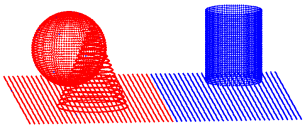
(a) Cloud 1



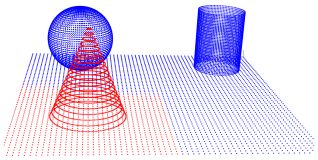
(b) Cloud 2



(c) Cloud 2 registered against Cloud 1 - Point-to-point ICP



(d) Cloud 2 registered against Cloud 1 - Point-to-plane ICP



(e) Cloud 2 registered against Cloud 1 - PFH ICP

Figure 3.15: Poor overlapping data sets needed to be registered (left) and after ICP registration (right)

$$\begin{bmatrix} \cos(30) & 0 & \sin(30) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(30) & 0 & \cos(30) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.866 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \\ -0.5 & 0 & 0.866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.31}$$

Table 3.17: Transformations matrix for poor overlapping data sets - Imposed transformation

$$\begin{bmatrix} 0.867 & -0.052 & 0.493 & 8.8 \\ 0.033 & 0.997 & -0.058 & 0.3 \\ -0.495 & 0.034 & 0.867 & 0.4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(a) Point-to-point ICP

$$\begin{bmatrix} 0.870 & -0.058 & 0.488 & 8.5 \\ 0.009 & 0.995 & -0.101 & 0.9 \\ 0.491 & 0.083 & 0.866 & 0.6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(b) Point-to-plane ICP

$$\begin{bmatrix} 0.867 & -0.058 & -0.494 & 8.8 \\ 0.033 & 0.997 & -0.058 & 0.3 \\ -0.496 & 0.034 & 0.867 & 0.4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(c) PFH ICP

Table 3.18: ICP matrices retrieved for noise-free partial data sets, rotated one by each other

In the following tests, a region of the original cloud has been extracted and rotated, with two different transformation, and then ICPs have been performed

	Point	Plane	PFH
Iterations	25	25	25
Time [s]	125	213	6.6
Relative Error	10^{-6}	10^{-20}	10^{-9}

Table 3.19: ICP algorithms comparison for scenes described in Figure 3.15 - Iteration Number Ending

	Point	Plane	PFH
Iterations	24	8	22
Time [s]	120	67	6.1
Relative Error	10^{-4}	10^{-4}	10^{-4}

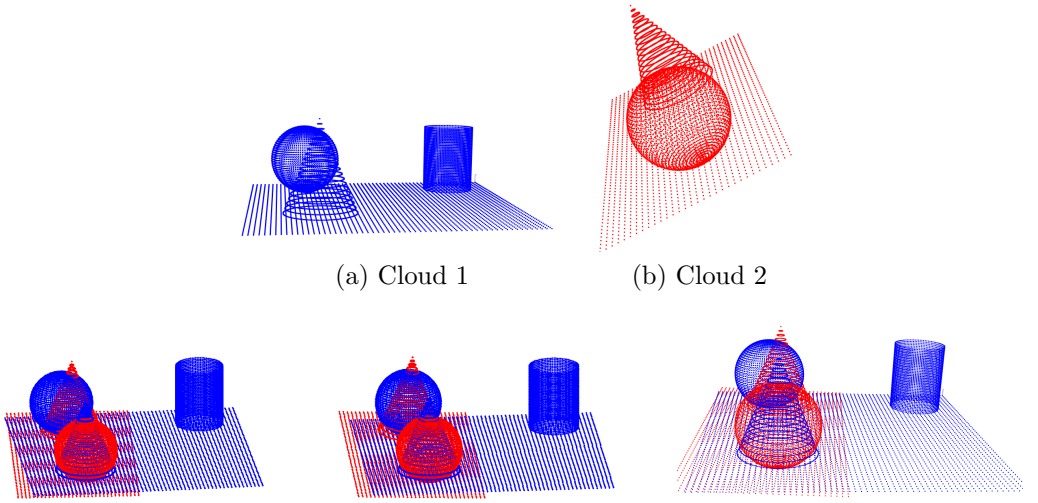
Table 3.20: ICP algorithms comparison for scenes described in Figure 3.15 - Relative Error Ending

retrieving a valid and non valid solution. In the first test the partial data set is rotated by 30° on the Y axes, like the very first test proposed in this section; in this case the response is correct for every algorithm, as visible from Figure 3.15 and Table 3.18. In the second the same partial data has also been further moved from the initial rotation and now neither algorithm can retrieve the correct transformation, as visible from Figure 3.16. As mentioned before, the results are comparable to the ones found in previous tests and PFH ICP is still the best choice, as it is at least ten time faster than the others, while reaching a sufficient level of accuracy. The results of the last test are displayed in Figure 3.16, and Tables 3.21, 3.22.

	Point	Plane	PFH
Iterations	25	25	25
Time [s]	126	211	10
Relative Error	10^{-4}	10^{-4}	10^{-3}

Table 3.21: ICP algorithms comparison for scenes described in Figure 3.16 - Iteration Number Ending

Also these last tests stress the importance of the initial positioning as the most critical factor for a proper recognition of the two data sets considered. Experimental tests to obtain the minimum percentage number of points that the data set have to share in order to achieve a correct registration (given a "good" transformation) found that this is value is around 50%. Under this value the ICP algorithms fail, due to their tendency to move one barycenter close to the other.



(a) Cloud 1

(b) Cloud 2

(c) Cloud 2 registered against Cloud 1 - Point-to-point ICP

(d) Cloud 2 registered against Cloud 1 - Point-to-plane ICP

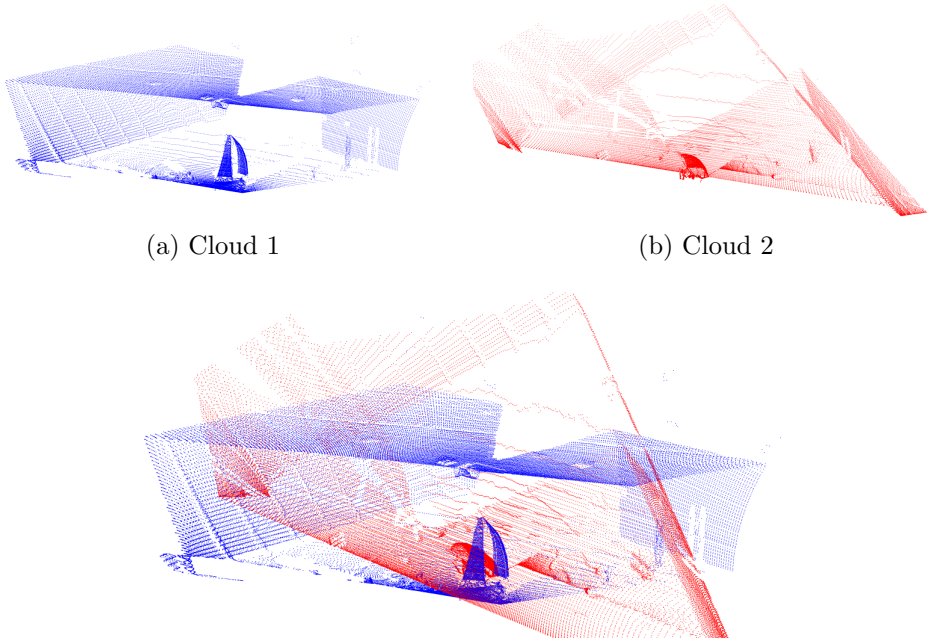
(e) Cloud 2 registered against Cloud 1 - PFH ICP

Figure 3.16: Poor overlapping data sets needed to be registered (left) and after ICP registration (right)

	Point	Plane	PFH
Iterations	22	10	29
Time [s]	110	85	12
Relative Error	10^{-4}	10^{-4}	10^{-4}

Table 3.22: ICP algorithms comparison for scenes described in Figure 3.16 - Relative Error Ending

Unfortunately this is the most likely situation to be encountered for the application considered. Each acquisition unit - with its proper coordinate reference system - scan a single sail, which means that for a complete acquisition of the rig a registration between two clouds is required. The scanners are placed properly in order to avoid occlusion, but this leads to a poor overlapping scanned area. A simple test has been conducted on raw acquisitions, the registration has been tried with the PFH ICP, since it has proven to be the algorithm with the best trade-off between time elapsed and accuracy. It fails in registering the raw clouds as shown in Figure 3.17, the procedure is not able to converge, even with a high number of iterations (more than 500 iterations).



(a) Cloud 1

(b) Cloud 2

(c) Cloud 2 registered against Cloud 1

Figure 3.17: PFH ICP Registration of wind tunnel acquisition

To overcome the problem a proper algorithm has been developed and presented in the last Section.

3.2.3 Registration of Wind Tunnel Acquisition

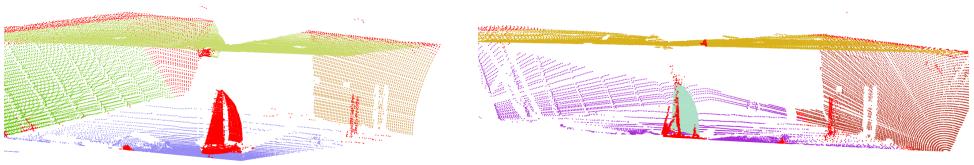
A new procedure has been proposed to achieve a correct registration of acquisitions taken in the wind tunnel. The first step is to compute a good initial alignment and eventually perform several iterations of the ICP algorithm, until reaching a desired level of accuracy. The rough alignment is not done through the SVD, but through a specific algorithm for this application, whose purpose is to reduce the imprecision due to points noise and not perfect correspondences. The key idea is to limit the number of points that has to be directly correlated and exploit the knowledge of the environment's composition. In particular, the presence the walls, which are inevitably visible in every acquisition, can be used to retrieve the transformation needed to register the point clouds.

Algorithm 3.2 Wind Tunnel Acquisitions Registration Procedure

1. Segmentation of the reference point cloud
   (a) Identification of the reference wall and reference ceiling [User Aided]
   (b) Extraction of the pose of the reference wall and reference ceiling
2. Segmentation of the registering point cloud
   (a) Identification of the registering wall [User Aided]
   (b) Extraction of the pose of the registering wall
3. First rotation : the registering wall is aligned with the reference wall
4. Segmentation of the wall-aligned point cloud
   (a) Identification of the registering ceiling
   (b) Extraction of the pose of the registering ceiling
5. Second rotation : the previously wall-aligned cloud is rotated again such that also the ceilings are aligned
6. Extraction of a pair of matching points from the reference dataset and the reference dataset, after the rotations [User Aided]
7. Translation: the selected points are overlapped, this transformation create a "roughly" aligned point cloud
8. ICP algorithm:
   (a) Filter the clouds in order to have comparable point clouds, removing non-overlapping areas
   (b) Perform ICP algorithm until reaching the desider accuracy level and store the resulting tranformation matrix
   (c) Apply the ICP transformation matrix to the registering cloud

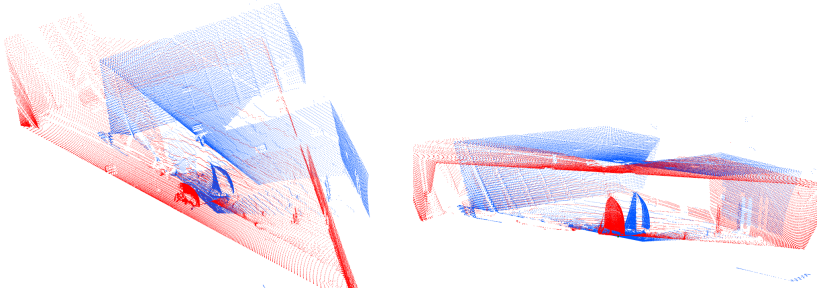
Using the entire set of points representing a wall allows also to reduce the inaccuracies coming from the laser scanner's sampling. To detect the planes in the scene the procedure uses two different segmentation algorithms, the RANSAC Paradigm and the Region Growing Clustering, that are explained in detail in section 3.3.2 and section 3.3.3 respectively. The procedure, fully explained in Algorithm 3.2, first aligns the walls of the two data sets, then the ceilings and, in the end, asks

the user to select a pair of corresponding points within the two clouds to perform a final translation.



(a) 1a- Identification of reference wall and ceiling

(b) 2a - Identification of the registering wall



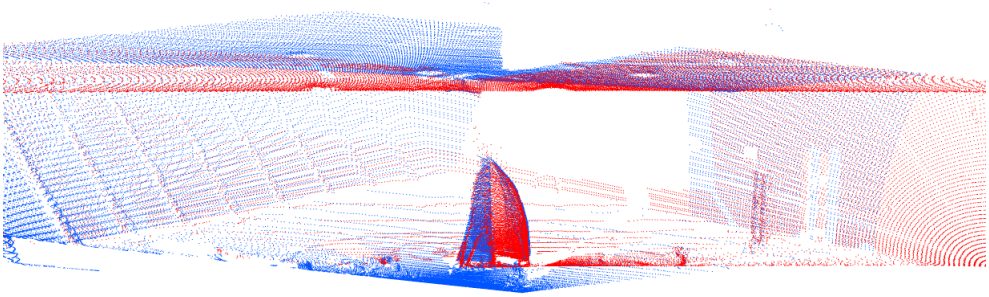
(c) 3 - Registering point cloud (red) after first rotation (top view)

(d) 5 - Registering point cloud (red) after second rotation (front view)

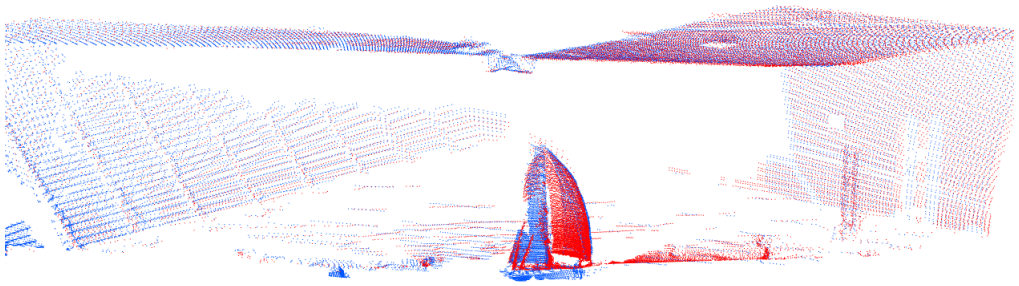
Figure 3.18: Wind Tunnel Acquisitions Registration Procedure - Step by step - Part 1

These steps allow to obtain a rough registration, which is although still influenced by the acquisition's sampling and the user's inaccuracy, via point picking. In order to reduce their unwanted contribution, the ICP algorithm is performed, until achieving a satisfying level of accuracy. The precision of the final transformation is evaluated through the Fitness Score of the algorithm, which is defined as the sum of squared distances from the source to the target. If its value is smaller than the acquisition accuracy found in section 2.4.3 then the registration is considered well-performed. Although limiting the user influence on the output accuracy, the procedure still needs for its interaction, especially in the planes recognition phase. This happens because the Region Growing algorithm, used for generating the cloud showed in Figure 3.18a and Figure 3.18b, is able to separate different objects in the scene, but can't recognize their geometry. Moreover, since the wall and the ceilings are both planar object and the two reference systems position and pose is not a priori known, developing a fully-automatic procedure to recognize them is extremely difficult and quite useless, compared to the simplicity of the user interaction.

The user have to recognize the wall and the ceiling, indicate it to the software,



(a) 7 - Registering point cloud (red) after the translation (front view)



(b) 8a - Clouds after the filter / Input clouds of ICP algorithms

Figure 3.19: Wind Tunnel Acquisitions Registration Procedure - Step by step - Part 2

selecting one of their points. The procedure is then able to identify their pose through the RANSAC paradigm, which can identify the normal associated to a plane. Given two vectors, v_1 and v_2, the transformation matrix, R, to rotate one onto the other come from :

$$v = v_1 \times v_2 \tag{3.32}$$

$$c = v_1 \bullet v_2 \tag{3.33}$$

$$R = I + [v]_x + [v]_x \frac{1-c}{\|v\|^2} \tag{3.34}$$

where $[v]_x$ is the skew-symmetric cross-product matrix of v:

$$[v]_x \coloneqq \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \tag{3.35}$$

The cross-product v between $v_1$and v_2 can be interpreted as the rotation axes, while the dot-product is the cosine of the angle that has to be applied. Figure 3.18c shows the result of the first rotation, which aligns the walls, and Figure 3.18d shows the rotation respect to the ceilings. The order of the transformations is not important, this means that the procedure returns the same output, when aligning first the walls and then the ceilings or vice versa. In the point picking phase for the translation, choosing points from the wall has proven to achieve a slightly better translation, respect to choosing them from the sails or ship model. Even if the clouds are already registered the result is not acceptable, as visible from the details in Figure 3.19a, the reasons for this misalignment are the same explained for the SVD algorithm in section 3.2.1. The ICP can't be performed on the clouds as they are at the current point of the procedure, because there are many areas where points do not match and this could cause the algorithm to fail, as demonstrated in section 3.2.2. To overcome this issue, a particular filtering has been implemented, starting from the classical spherical filter.

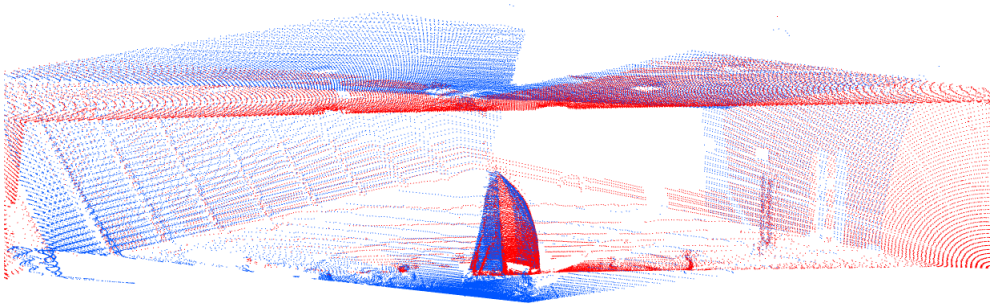
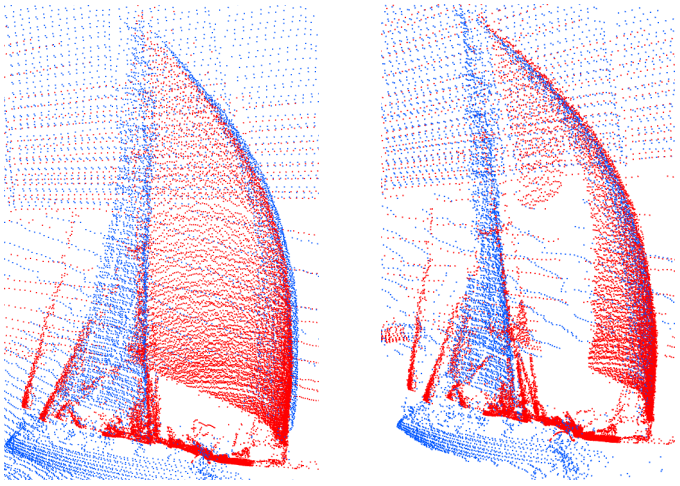


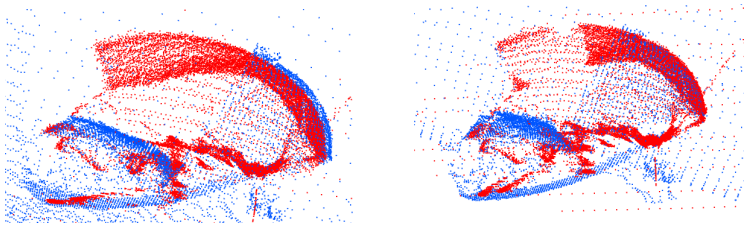
Figure 3.20: 8c - Registering point cloud (red) after the ICP algorithm (front view)

Normally the spherical filter create a sphere of fixed radius around each point, if the sphere does contain no other points from the same clouds (or less then a fixed number of points), the center point is considered as noise and removed from the data set. In this case the filter, after constructing the sphere, checks if there are close points from the other data set; the point clouds filtered are shown in Figure 3.19b. Note that, correctly, all the areas where the data do not overlap are removed,

especially on the ceiling and in the background, but also part of the sails, from each acquisition as been canceled, because there are no correspondence. Without the use of this filter the ICP can't converge to a solution, but it should be larger enough to keep the majority of the sails point. The final registration scene is showed in Figure 3.20.



(a) Gennaker Sail - Side View



(b) Gennaker Sail - Top View

Figure 3.21: Registering point clouds details - pre-ICP (left) and post-ICP (right)

Although the scene does not seem to change respect to Figure 3.19a, there are improvement, as visible from Figure 3.21, where close-ups of the clouds before and after the ICP are compared. The effectiveness of the algorithm can be proven also through a statistical analysis the minimum point to point distance changes between the registering cloud and the reference cloud with, or without, the last refinement step. Figure 3.22 shows the histogram of the distribution of the registering points respect to their minimum distance with points belonging to the reference cloud.

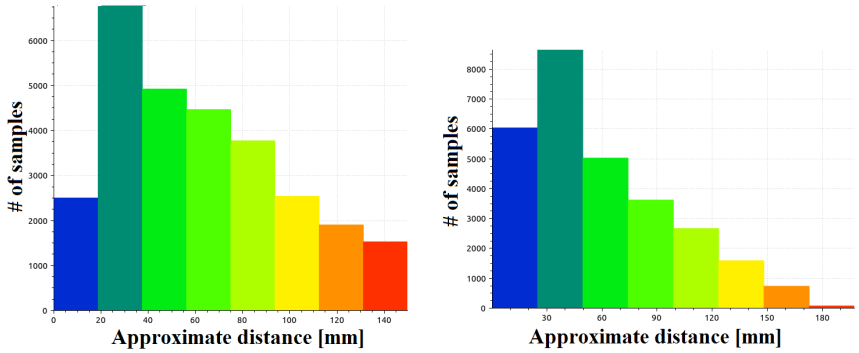


Figure 3.22: Statistical point to point distance before (left) and after (right) performing ICP

As visible, the pre-ICP cloud has a steadier distribution, while the post-ICP distribution presents a higher number of samples, more than doubled, in the low-distance (blue) zone. Evaluating the performance of the whole procedure is not easy task, because the output of the ICP implementation, the fitness score, the sum of squared distances from the source points to the target points, in the case of clouds with different points can't be used to retrieve the single point registration accuracy. A generic, mean accuracy can be found dividing the fitness function with the total number of points of the registering data set:

$$\bar{f} = \frac{F_n}{n} = \frac{2345.3[mm^2]}{32829} = 0.071[mm^2] \tag{3.36}$$

which represents the medium area for a registering points to contain a neighbor reference point. This result is not significant of the procedure precision, since we are comparing clouds that do not have point to point correspondences and moreover, still after the filter, there are parts that do not overlap (see Figure 3.19b). To test the effective accuracy of the procedure a wind tunnel full acquisition is rotated by 90° one of the axes and then moved by 1000 mm in each direction is registered against its original position and pose. The imposed transformation matrix is compared to the registration matrix returned from the procedure in (3.23b).

The registration is almost perfectly achieved and the imperfections can be imputed to numerical approximation in the transformation implemented. The overall fitness function is $F_n = 9.8 \cdot 10^{-4}$ mm^2, which is expected, as the clouds considered are composed by the same, identical points.

Since the overall fitness is well below the acquisition system's accuracy (section 2.4.3), it follows that the single point fitness has to be smaller and that the condition

$$\begin{bmatrix} 1 & 0 & 0 & 1000 \\ 0 & 0 & -1 & 1000 \\ 0 & 1 & 0 & 1000 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0.999 & 10^{-8} & 10^{-7} & 1000 \\ 10^{-7} & 10^{-7} & -0.999 & 999.97 \\ 10^{-7} & 1 & 10^{-7} & 1000.01 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(a) Imposed transformation

(b) Retrieved transformation

Table 3.23: Wind tunnel registration with perfectly matching clouds

to obtain a well performing procedure is satisfied.

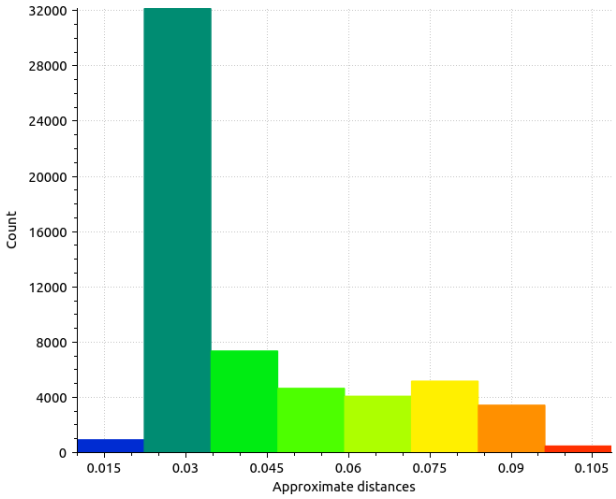


Figure 3.23: Statistical point to point distance of the registration a same transformed cloud against the original cloud

After achieving the registration, the final transformation matrix is saved as an output file, which can be applied to other acquisitions from the same set that need to be registered, saving time in the complete processing operations.

3.3 Scene Interpretation and Clustering

Segmentation and clustering are generally described as the operation of dividing large data sets in smaller, disjointed subsets.

Handling chunks of data instead of the whole has many advantages, but the developing of these algorithms was first pushed by the need of reducing the processing time of further operations (i.e. triangulations). The initial attempts were carried on only considering the euclidean distance as discriminating factor, while eventually adding additional information, like normals and curvature allowed to achieve a more intuitive decomposition.

Segmentation and clustering end up with almost the same result, although with a main distinctive difference, as segmentation supply additional information on the subsets returned, while clustering only creates several possible data sets. Segmentation returns extra information since it searches in the data set pre-determined models, such as spheres, planes or cones and it can identify the parameters of the model, like the radius of the sphere. On the other hand clustering divides the original data in subsets starting from its local properties, so it could be run without the knowledge on what shape has to be searched.

Since the difference between the output of the two procedure is so subtle that in many situations they can easily be used interchangeably,this work refers to "segmentation" as a general procedure that extracts a set of points from a wider data set.

Segmentation is a key process in computer vision and its main contribution goes to obstacle avoidance problems, where identifying information on the position of an object, its spatial coordinates and its dimensions are clutch towards the resolution of the issue. Most of the algorithms developed were initially applied on 2D data coming from cameras. Later, with the spread of 3-D sensors (laser scanners and depth cameras), they started have being applied to point clouds.

In this chapter a comparison of different algorithms is presented. Strength and weak points are discussed, in order to establish the most suitable for the application considered: it has to extract correctly the data points from the whole scene acquired.

In detail, the Euclidean Cluster Extraction algorithm is presented at first, since is the simplest and

Then segmentation procedures using RANSAC and Region Growing algorithm are described and finally the Hough Transform is reported. These procedures were

tested on simulated data, depicting simple scenes with regular geometry objects, and then on real acquisition scenes with both mainsails and gennakers.

3.3.1 Euclidean Cluster Extraction

As mentioned before, this procedure divides the original scene into parts, creating many subsets whose cardinality is way smaller than the original one. Euclidean Cluster Extraction, as the name suggests, relies only on the euclidean distance to populate the clusters. This means that given two points belonging to the data set, they are put in the same cluster only if their distance is smaller then a given threshold. As it is possible to imagine, rarely this kind of clustering is able to separate some objects, because when the rate of acquisition is constant, which is an expected choice, consequently the point-to-point distance is locally fixed. What this algorithm is capable of doing is to split wide pieces from the original data, for example it could be useful for separating a target from a further background or similar application where the area of interest is limited and well localized.

To populate a cluster, the Euclidean Cluster Extraction first selects a point p, labeled as "seed", in the data set and then search for all the other points within a sphere of radius r centered in p, which will form the initial cluster. Then these added points turn into seeds and the procedure goes on until no more points can be added to the current set. The pseudocode of the full algorithm is shown in Algorithm 3.3.

A cluster seed is the first concept that has to be introduced. It acts as a starting point from which the algorithm develops and can be chosen randomly, as in this case, or has to satisfy some conditions in order to be picked. In the procedure, a K-d tree (see section 3.1.7 for details) is exploited in the neighbors research, as its space-partitioning data structure for organizing points is able to drop significantly the computational costs. The only parameter that can be handled is the radius of the sphere, which enlarges or shrinks the volume that is considered in the close points research. Setting the radius to a value similar to the density of the cloud will result in a finer research, while, on the contrary, setting it to a high value (w.r.t. the cloud density), will split the data set in a coarse way. This is the main flaw of the Euclidean Cluster: an acquisition of a real scene will unlikely have a defined distinction between the objects, because also intermediate points are acquired, and so the procedure will progressively incorporate them in one cluster. So a scene

Algorithm 3.3 Euclidean Cluster Extraction

1. Create a K-d tree representation for the input point cloud dataset $\mathbf{P}$;
2. set up an empty list of clusters $\mathbf{C}$, and a queue of the points that need to be checked $\mathbf{Q}$;
3. for every point $\mathbf{p_i} \in \mathbf{P}$, perform the following steps:
   (a) add $\mathbf{p_i}$ to the current queue $\mathbf{Q}$;
   (b) for every point $\mathbf{q_i} \in \mathbf{Q}$ do:
      i. search for the set $\mathbf{P_k}^{\mathbf{i}}$ of neighbors point in a sphere with radius $\mathbf{r} < \mathbf{d_{th}}$;
      ii. for every neighbor $\mathbf{p_i}^{\mathbf{k}} \in \mathbf{P_k}^{\mathbf{i}}$, check if the point has already been processed, and if not add it to $\mathbf{Q}$;
   (c) when the list of all points in $\mathbf{Q}$ has been processed, add $\mathbf{Q}$ to the list of clusters $\mathbf{C}$, and reset $\mathbf{Q}$ to an empty list;
4. the algorithm terminates when all points $\mathbf{p_i} \in \mathbf{P}$ have been processed and are now part of the list of point clusters $\mathbf{C}$.

composed as in Figure 3.24 (left) it is possible to use this procedure to identify the objects, while in Figure 3.24 (right), the identification is no more achieved.

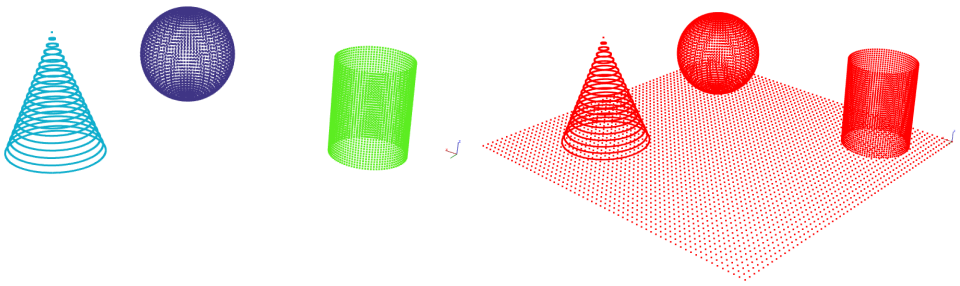


Figure 3.24: Comparison between a scene composed by the objects (left) and one in which the various objects are more realistically placed on a plane (right) - Different colors refers to different clusters.

Due to this problem it is evident that for the application considered in this work the Euclidean Cluster Extraction is not a suitable solution for the identification of the sail cluster. Methods that take into account more geometrical information are needed.

3.3.2 RANdom SAmple Consensus paradigm - RANSAC

The RANSAC algorithm starts from the assumption that in the data considered there is a subset of points that can be explained through a mathematical model. Data can then be divided in inliers, i.e. points that well suit the considered model (accepting a certain tolerance), and outliers, i.e. points that do not suit it. The original paradigm was developed by Fischler and Bolles in [Fischler and Bolles, 1981] to solve the Local Determination Problem related to cartography. The goal was to find the spatial position from which an image had been shot evaluating the perspective distortion between straight lines joining some know landmarks. In this case the model to be found was the simplest one, the straight line. Afterward, other authors improved the procedure including some planar models such as circles and ellipses. Finally, the first three-dimensional implementation was published by Schnabel, Wahl & Klein in [Schnabel et al., 2006], they were able to identify planes, spheres, cylinders and cones.

Despite the fact that the yacht sails could not be represented perfectly by these regular geometries, the RANSAC algorithm was taken into consideration to test whether, under certain tollerance, it could extract the sail shape, approximating it with a cone or a cylindrical model. The results for these tests are reported below.

The algorithm could be divided in three main parts. In the first step, a subset containing the smallest number of points required to construct the model sought is extracted (and the model is effectively created); in the second step, data points are tested and flagged as inliers or outliers with respect to an error tolerance. Repeating these steps leads to different identified shapes, each formed starting by different minimal subsets. The final step is evaluation of the best shape extracted, that is the shape that provides the maximum number of inliers, i.e. the highest score function. The RANSAC algorithm, reported in Algorithm 3.4, requires as input only the model describing the sought shape and a parameter ε representing the maximum discrepancy acceptable between a point and the model constrain. It returns only one cluster for each data set; so that, to extract two or more objects in a scene, an external iteration loop is needed.

In the next sections, the procedure steps are discussed more in detail.

Algorithm 3.4 RANSAC paradigm

1. Computation of an octree representation for the input point cloud data set **P**
2. Extraction of the minimal point set **S** to create the model
3. Creation of the candidate shape ψ from the points in **S** and computation of the shape's parameters (i.e. radius, planarity...)
   (a) Identification of inliers or outliers points from the point cloud **P**
   (b) Evaluation of the score function $\sigma(\psi)$
   (c) Re-computation of the shape parameters starting from all the inliers identified
4. Repetition of step 2 and 3 for a fixed number of times to extract many candidate shapes and their score functions
5. Rejection of candidate shapes with too few points
6. Extraction of points of **P** belonging to the shape with the highest score function

3.3.2.1 Extraction of the minimal point set

RANSAC algorithm uses both information about the spatial coordinates of the points and their normals, that are computed implicitly within the algorithm. Whatever the shape to identify, the algorithm extracts a minimal subset composed of three points, with their corresponding normals, $<(p_1,n_1),(p_2,n_2),(p_3,n_3)>$. The first point p_1 of the minimal subset is picked randomly in the whole data set. Then, after computing an octree spatial subdivision - extremely useful in the research of neighbor points, (see section 3.1.8), a box C (or a level of the tree) containing p1 is selected. p_2 and p_3 are chosen within the same box C. The algorithm, in fact, works under the hypothesis that the shape sought is a local phenomenon in a wider set of data and so a shape created from close points has more probability to be the right one instead of one built from distant points. Moreover, choosing C from a proper level of the octree is an important aspect. The smaller the C dimension is, the greater the probability that the points in the minimal subset correspond to the same shape. For sure, the density of the cloud must be taken into account and the algorithm can automatically adapt to point cloud with non-uniform density, which is especially convenient in this study.

3.3.2.2 Shape estimation

Each model is constructed differently and, if possible, is immediately checked for consistency, in order to avoid to carry on shapes that have low possibility of being the correct one.

Planes <p_1,p_2,p_3> constitutes a set that permits to retrieve the parameters of a plane. If the deviations of the normals n_1, n_2, n_3 from the plane's normal is less than an angle threshold, α ,then the shape is accepted, otherwise it is immediately rejected.

Spheres a sphere is fully defined by two points with their normal vectors. The sphere is accepted if all the three points are within a distance of ε from its surface and all their normals do not deviate by more than α degrees.

Cylinders to generate a cylinder two points and the corresponding normals are needed. Again the shape is verified by applying the thresholds on euclidean distances and normal deviations.

Cones although a cone can be constructed with the same starting base as the previous two shapes, for simplicity all the points and normals are used in its generation.

The procedure to reconstruct each model and its parameters is explained in detail by Schnabel, Wahl & Klein in [Schnabel et al., 2006].

3.3.2.3 Score Function

The score function $\sigma(\psi)$ is responsible for measuring the quality of the shape ψ and is formally defined as $\sigma(\psi) = |P_\psi|$, where $P_\psi$is the maximum connected component on the parameter domain of the corresponding shape and can be expressed with:

$$P_\psi = maxcomponent(\psi, \hat{P}_\psi)$$

$$\hat{P}_\psi = \{p | p \in P \wedge d(\psi, p) < \varepsilon \wedge arccos(|n(p) \cdot n(\psi, p)|) < \alpha\} \tag{3.37}$$

In (3.37) the second condition represents the threshold on the euclidean distance from the point to the shape, while the third is the correspondent threshold on the deviation between their normals. The connected component is sought in a bitmap

in the parameters space where a pixel is set if a point is projected into it and the parametrization changes depending on the model considered. In the end what stands out is that the score function is simply a cardinality operator and due to this the RANSAC algorithm will return, from the candidate shapes found, the one which is able to explain the higher number of points; this could lead to some data misinterpretation, that will be shown in the Result section. Moreover there is another adjustment to explain, which take place in the final step of 3.4 and it's the fact that the score function is not calculated over the whole data set, because this operation would be really computational expensive, so the expected value of $\sigma(\psi)$ is forecasted. Before calculating the connected component, the whole data set is split in disjointed random subsets $P = \{S_1 ... S_r\}$, the score is evaluated only on $S_1$and and estimate of the score is calculated through inferential statistics as: $\hat{\sigma_p}(\psi) = -1 - f(-2 - |S_1|, -2 - |P|, -1 - \sigma_{S_1}(\psi))$. f(N,x,n) is the mean plus/minus the standard deviation of the hypergeometric distribution, so $\hat{\sigma_p}(\psi)$ can assume two values, a or b. $\hat{\sigma_p}(\psi)$ can be seen as a confidence interval [a,b] for the true score $\sigma_p(\psi)$, whose expected value $E(\sigma_p(\psi))$ is given by $\frac{a+b}{2}$. At this point each shape has an expected value for its score, but it comes with an uncertainty, that has to be taken in consideration, before comparing the values to extract the best candidate. For this reason $E(\sigma_p(\psi))$ for a candidate is accepted only if no confidence interval of others overlaps with it. If this happens, then the estimate $\hat{\sigma_p}(\psi)$ is recalculated for all, adding one more subset S_i and so on until the uncertainty in the estimation decreases, the intervals do not overlap and they can be compared.

3.3.2.4 Parameters

Here is a recap of the parameters involved in the RANSAC paradigm and the differences involved between the theoretical algorithm explained and the implementation using the PCL libraries.

- **Minimum size** τ of a candidate shape; used to eliminate candidates too small. Not settable by the user in the Random Sample Consensus model from the PCL, so either it's set automatically starting from the size of the original point cloud or this supplementary control is not used.
- **Distance error tolerance** ε, establish what is the acceptable distance from a point to the hypothetical surface so that the point is considered as an inlier. Should correspond as possible to the accuracy of the scanning device. This is

the only parameters that the PCL use as input, due to the strict dependence to the acquisition.

- **Normal deviation tolerance** α, the least critical parameter, as said in [Schnabel et al., 2006], and in general can be set arbitrary large, even though smaller values permits the find smoother edges. However this can be also easily done in a post-processing step, so setting α to a high value helps decreasing the computational cost of RANSAC, without any drawback. PCL does not consider it as input.

- **Pixel-size** β, used in the passage to the parameters space in the calculation of the score function. Should be equal to the sampling density of the point cloud. If this is unknown, it can be replaced by the result of a minimum neighbors computations on the data, which is what is done in the PCL method. In case of point clouds with variable density, the minimum local density should be returned by the nearest neighbors procedure.

In addition it has to be reminded that, since the normals are computed inside the method, in the implementation we don't have access to their parameters, that are explained in section 3.1.3.

3.3.2.5 Results

In this section segmentations of various scenes are presented, they are constructed through mathematical models, ideals or with the addiction of noise, to evaluate the actual performances of RANSAC and identify its main problems. A reconstructed scene is set up, with all the basic elements available, built from code, which means that there is a full three-dimensional shape and its points are well spaced, meaning that every area of the shape is well acquired. Also actual scenes will be used in the test, both similar to the previous and segmentation of the sails will be performed.

Let's begin with showing some good segmentations, for each model available and their statistics.

As visible all the basics shapes are detected with a high level of accuracy, as the percentage of correctly identified points is around 90% for all, with the notable exception of the plane, which takes in also points from the other shapes. The time needed by RANSAC to complete the procedure rises with the increase of the number of parameters that have to be identified, as expected.

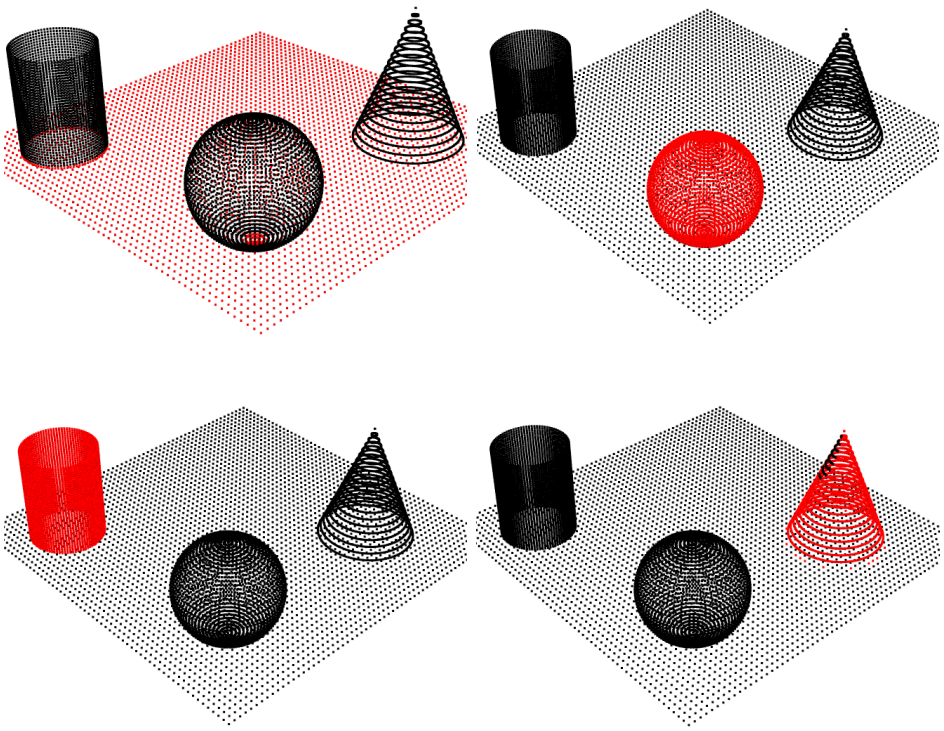


Figure 3.25: Basic elements segmented with RANSAC paradigm - Up left: plane, up right: sphere, down left: cylinder, down rigth: cone

Unfortunately a good segmentation strongly depends on an suitable value for ε, because a relatively high value could lead to a total misinterpretation of the data. The main problem in RANSAC is that it tries to force the input data into the selected model. It returns the candidate shape with maximum cardinality, as explained, without further considerations, so could happen that in the scene a subset of data can create a candidate shape with high score function, even though it does not really form a complete shape. In Figure 3.26 a failed segmentation of a cone, with ε=0.5, is showed. Similar results have happened also with cylinders and especially with spheres, in scenes with a sphere on a larger plane; in this case the plane could be identified as a sphere with a huge radius. A strong dependence on the geometric composition of the scene has been detected, because the relative position of the object on the scene also influenced the interpretation.

With the presence of a realistic noise the situation worsen to the point that the cone and the cylinder can't anymore be identified, however the effect of the

Shape	# of original points	# of identified points	% of correctly identified points	ε	Time Elapsed [ms]
Plane	3600	4140	85	0.04	202
Sphere	8100	8105	99.94	0.04	31
Cylinder	3600	3608	99.78	0.04	1159
Cone	3620	3301	91.71	0.04	8444
SCENE	18920				

Table 3.24: Segmentation of basics elements with RANSAC paradigm

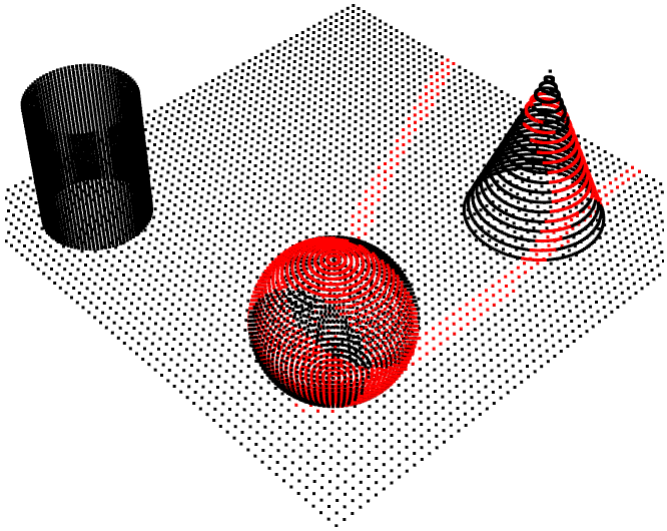


Figure 3.26: Failed segmentation for a cone

inaccuracies can still be countered raising the value ε on the sphere, as visible in Figure 3.27, where on the left the inliers band is narrower and on the right it's wider and allows all the points to be grouped in one shape. One of the identification case that remains unclear is the result of a segmentation where the candidate shapes have the same cardinality, a rare case to be honest, but still worth of a more exhaustive analysis. To study this situation a scene with a four cones, built with the same number of points, but with different geometrical dimensions, is set up.

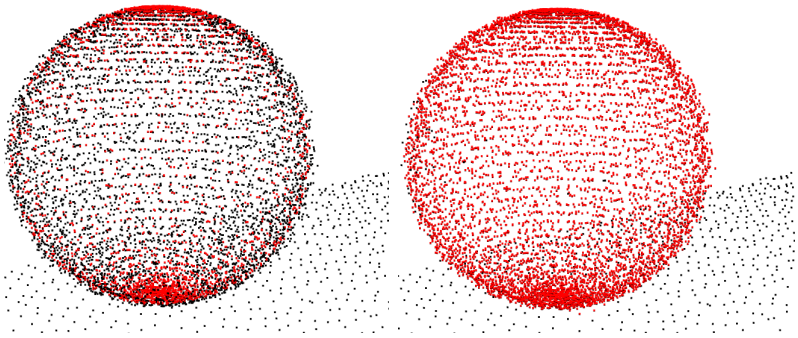
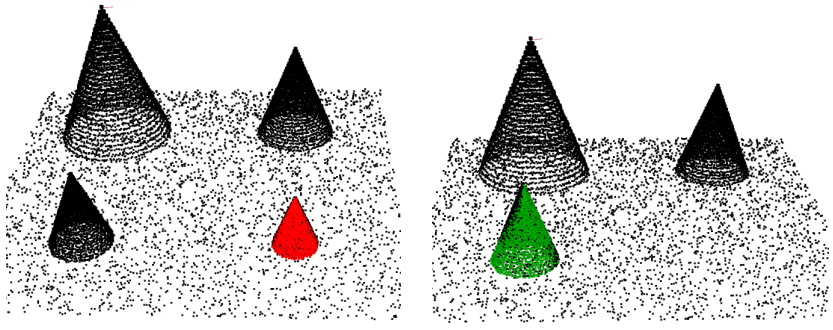
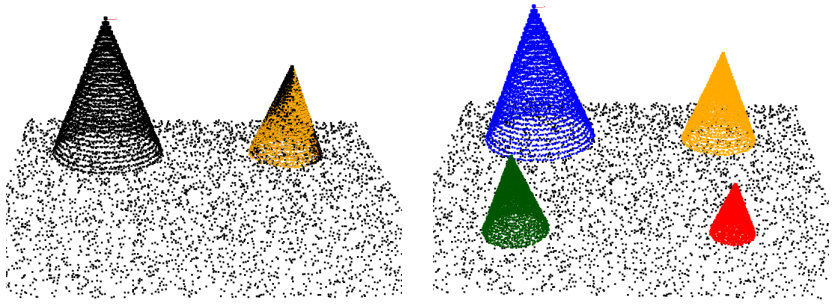


Figure 3.27: Noise effect and counter-effect of ε (on the right ε=0.04, on the left ε=0.3)



(a) Iteration #1 (b) Iteration #2



(c) Iteration #3 (d) Iteration #4

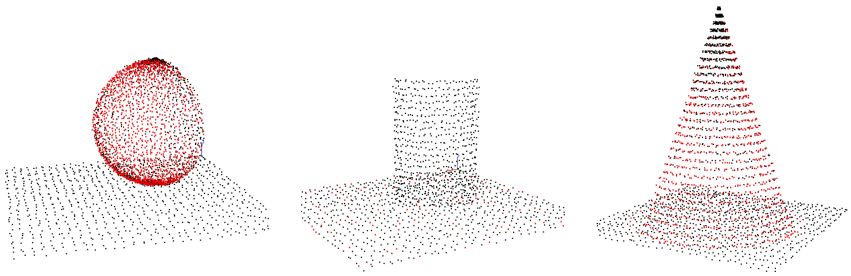
Figure 3.28: RANSAC on shapes with same number of points but different geometric dimensions: smaller shapes are identified first

When a shape is found then it is removed from the point cloud and then the algorithm is run again to confirm the result. Figure 3.28 shows each step of the iteration and what stands out is that, on other conditions being equal, RANSAC first identifies the smallest set.

Until this point all the figure used in the tests are fully three dimensional, but

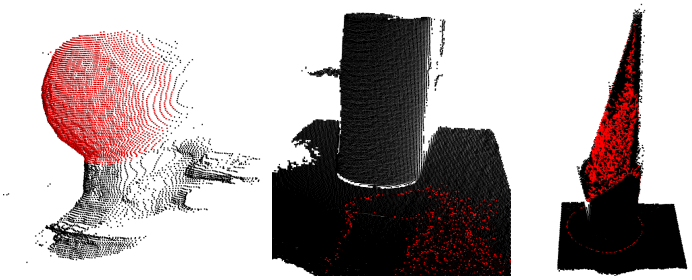
this does not represent a real acquisition, which has a fixed field of view.

In the best case a single point cloud can cover only up to 180 degrees around an object. Moreover the attempt or retrieving a sail through this procedure supposed that its surface could more or less be approximated as a part of a cylinder or cone, so investigating the performances of the algorithm with partial shapes could bring some information on the possibility of a successful conclusion.



(a) Sphere (b) Cylinder (c) Cone

Figure 3.29: Identification of partial shapes, only $\frac{2}{3}$ of a shape are available

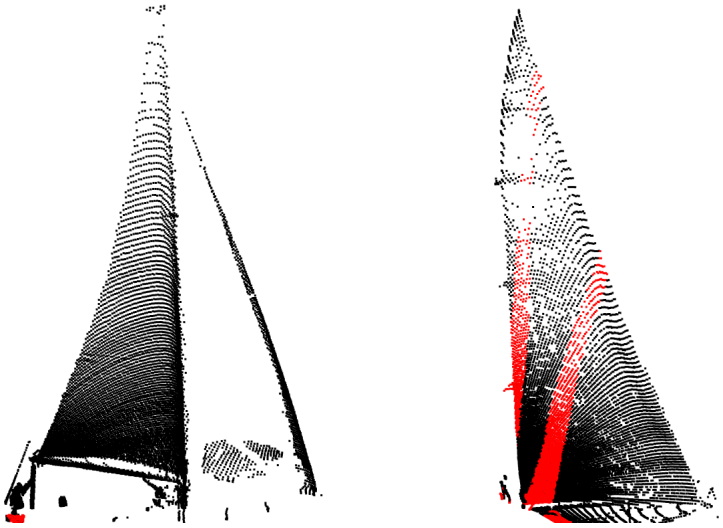


(a) Sphere (b) Cylinder (c) Cone

Figure 3.30: Identification of shapes coming from real acquisitions

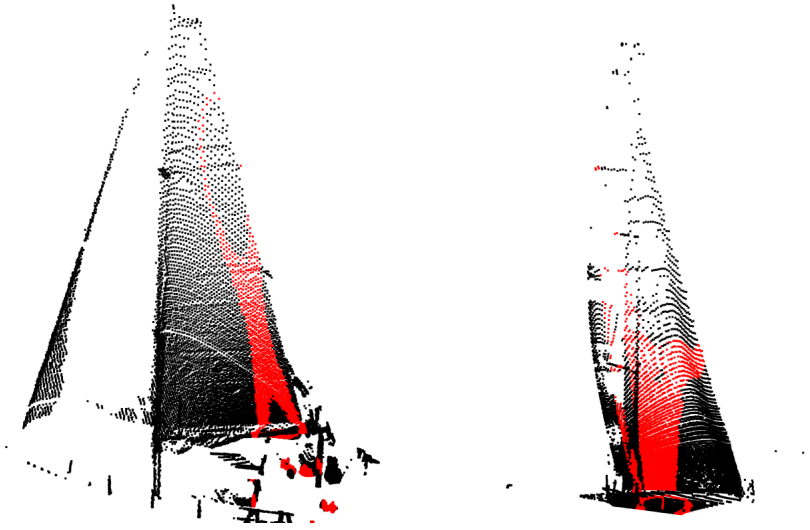
In Figure 3.29 is showed that the algorithm has a mixed response, because it can successfully retrieve the shape of a 60° cone and sphere, but fails in the cylinder case. Similar observations can be made on model coming from real acquisitions, as can be saw in Figure 3.30. This series of point clouds were registered with a Kinect unit instead of the acquisition system described and characterized in section 2.4.3. The last step is to perform the segmentation on real acquisitions taken both on the field and in the wind tunnel. The value of ε has to be chosen considering the type of data used, because the laser scanner has a spatial definition that changes with the distance of the surface acquired; a worst-case value is considered initially and

then adapted to the data set. The images show the best results that is possible to obtain through the RANSAC and, as can be seen, it's absolutely ineffective. The tables 3.25 stress out the great difference from these segmentation from the first performed.



(a) Mainsail (b) Gennaker

Figure 3.31: RANSAC segmentations of cylinders on sails acquisitions, $\varepsilon = 60$



(a) Mainsail (b) Gennaker

Figure 3.32: RANSAC segmentations of cones on sails acquisitions, $\varepsilon = 60$

	# of sail points	% of correctly identified points
Mainsail	7054	0
Gennaker	11799	28.86

(a) Cylinder

	# of sail points	% of correctly identified points
Mainsail	6537	21.65
Gennaker	8837	43.7

(b) Cone

Table 3.25: Segmentation of sails through RANSAC

3.3.2.6 Conclusions

The RANSAC paradigm fails to identify the sails in every case tested, the reason is that the sail considered not only are curved but also a little twisted and so they can't be approximated as cones or eventually cylinders. An effective use of the algorithm remains restricted to scenes where its main hypothesis is verified, where it showed great precision in first analysis. The objects searched have to be really one of the basic shape; given this the algorithm provided great adaptability in presence of considerable level of noise and is able to detect also partial objects. Another main drawback is the presence of an error threshold, ε, which has to be adapted to the data set, in order to have a significative result. The RANSAC paradigm is a good choice for segmenting scenes composed of quasi-geometrical shapes or when a single object has to be decomposed in its principal parts (i.e. identifying its faces) but in this case, another solution has to be found.

3.3.3 Region Growing Clustering

The region growing type of solution was developed initially for the segmentation of intensity images and starts from the postulate of similarity of pixels within regions. This principle states that pixels that represent the same region or object share the same properties, like the same level of gray in a gray-scale image. This type of procedure was particularly improved by Adams and Bischof in [Adams and Bischof, 1994], where they proposed a Seeded Region Growing algorithm, in which they required an input set of seed points. As deducible from its name the algorithm proceeds to create a region with similar properties starting from each seed. Since it divides the data set considering the information coming from

local properties and not from an external model, the region growing is a clustering algorithm.

Algorithm 3.5 Region Growing Clustering

1. create a region list $\mathbf{R}$, initially empty, an available points list $\mathbf{A}$, such that $\{\mathbf{A}\} \leftarrow \{1, ..., |\mathbf{P}|\}$ and a seed list $\mathbf{S}$;
2. while $\mathbf{A}$ is not empty:
   (a) create a current region set $\mathbf{R_c}$;
   (b) select the seed as the point with minimum curvature in $\mathbf{A}$ and add it in $\mathbf{S}$;
   (c) for i=0 to size($\mathbf{S}$) do:
      i. find the nearest neighbours $\mathbf{B_c}$ of the current seed point $\mathbf{S_i}$;
      ii. for j=0 to size($\mathbf{B_c}$) do:
         A. for the current neighbour point $\mathbf{P_j} \leftarrow \mathbf{B_c}(j)$;
         B. if $\mathbf{P_j}$has still not be assigned ($\mathbf{P_j} \in \mathbf{A}$) and
         $$cos^{-1}(|N\{\mathbf{S_i}\}, N\{\mathbf{P_j}\}|) < \theta_{th}$$
         then add the point to the region and remove it from the available points:
         C. if
         $$c(\mathbf{P_j}) < c_{th}$$
         then add the point to $\mathbf{S}$;
   (d) if the current region $\mathbf{R_c}$ has too little points or too much points ($|\mathbf{R_c}| < N_{min} \wedge |\mathbf{R_c}| > N_{max}$) then is rejected, else
   (e) return the current region $\mathbf{R_c}$ as cluster;
3. return the set of clusters.

The transposing in the three dimensional space was implemented by Rusu in [Rusu, 2010] as a "natural extension" of the Euclidean Cluster Extraction. His first intention was to identify and extract different surfaces colliding with each others, like a table lean on a wall, so he has chosen as the discriminating factor the normal of a point, which replaces the Euclidean distance used in section 3.3.1. Moreover he chooses the automatically pick the seeds as the points with minimum curvature in the original data set, while all the previous procedures required them as inputs.

This is quite important because it means that the segmentation starts from flat areas and proceeds to aggregate as many points as possible, according to their properties, as explained in Algorithm 3.5. When considering if adding a point to a cluster it is compared with the current seed in consideration: if their normals does not diverge more than a threshold ϑ_{th} then they are considered as part of a single object. Here the critical condition for a region growing procedure is checked, if the point has a low curvature, or at least lower then a curvature limit c_{th}, then is added as a seed point for the current considered cluster.

This step allows the grow gradually the number of points that can be added; consider three points p_1, p_2, p_3 and their normals n_1, n_2, n_3, everyone on a curved surface, such as n_3 diverges from n_1 more than ϑ_{th} and instead n_2 that has the opposite property both respect to n_1 and n_3. p_3 can be added to the cluster of p_1 only through p_2, while it wouldn't be added at all with a different type of clustering. Through the mechanism it is also assured that the region grows from the areas with lower curvature, in other words planar surfaces will be identified first. The Region Growing Clustering is much simpler algorithm respect to the RANSAC because it needs less objects and computations to create a cluster; moreover the user has more control on the performance, as a higher number of parameter is available to be set. Another peculiarity is the possibility for a point to not be part of a cluster at the end of the steps, which happens when it is assigned to a cluster that is eventually rejected due to its cardinality.

3.3.3.1 Parameters

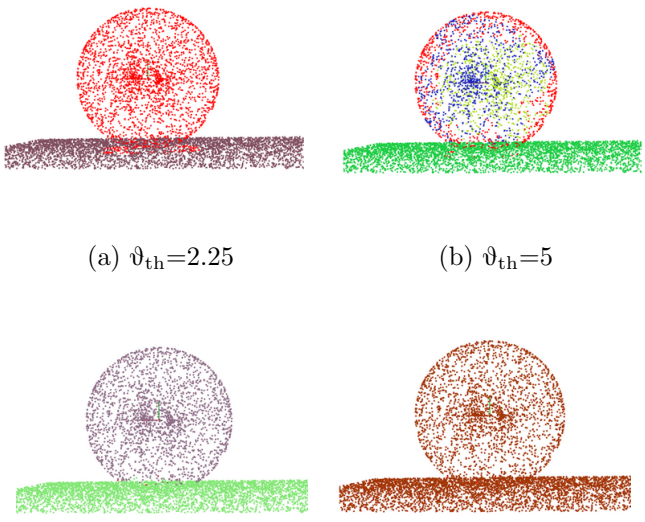
The final clustering if the data set depends from many parameters, listed below:

- **Minimum size $\mathbf{N_{min}}$** of the cluster: allows to reject clusters that have too many few points. Although its simplicity it's a very useful parameter, especially in this work, where the the sail usually takes up to one third of the total points of the whole data set and never less then one tenth. We can tune it in order to try to achieve a unique cluster containing the sail, but this is effective only with data sets coming from on field acquisition, since they don't have a background, which is present in data sets from the wind tunnel. Walls and ceiling are identified and most of the times are represented by a number of points comparable to the one of the sails.

- **Maximum size $\mathbf{N_{max}}$** of the cluster: allows to reject clusters that have too

many points. It is the complementary parameter of N_{min} and it could be used in the wind tunnel acquisition to eliminate possible clusters containing the wall points, while in other cases it's quite useless, because the sail is usually the cluster with the most points.

- **Smoothness threshold ϑ_{th}**: allows a point to be added to a cluster if its normal differs less than it from the normal of the current seed, so increasing the smoothness threshold allows points with large normals difference to be part of the same cluster. The main advantage of this parameters is that it is



(a) ϑ_{th}=2.25 (b) ϑ_{th}=5

(c) ϑ_{th}=12 (d) ϑ_{th}=20

Figure 3.33: Smoothness threshold effect on a sphere generated with random points - red points can't be assigned to a cluster, other colors correspond each to a different cluster. Other parameters: c_{th}=50 , N_{min}=50 , N_{max}=1000000.

independent from the point cloud dimensional scale, so it is possible to compare values from an outdoor acquisition with the ones used in a segmentation of a wind tunnel acquisition. On the other hand it has to be adapted not only to the type of sail considered, but also to the sailing point of the boat at the registration of the sail shape. The mainsail has usually an almost flat surface, while gennakers are more convex and close-hauled andature generates straighter sails, on the contrary running before the wind creates more rounded shapes. To explain the mechanism, a segmentation on a simple scene as been performed, where the other parameters are set to be non influential. Figure

3.33 shows the creation and composition of the clusters with the change in the value of ϑ_{th}. Points colored in red are the ones that are unassigned to a cluster, as explained further above, while in Figure 3.34 there is a detail of the normals calculated. Starting from the up left, in Figure 3.33.a the only object returned is the plane, because all of its points have normals that are very similar and fall within the angle threshold. With the rise in the value of ϑ_{th} (see Figure 3.33.b) two different clusters are generated, although they can't unite because the normals generated from random points can be noisy and lead to this issue. However this is solved in Figure 3.33.c, where an adequate level for the threshold allows to successfully separate the two objects and an additional rise would group all the point in a single cluster, since no points is being filtered anymore.

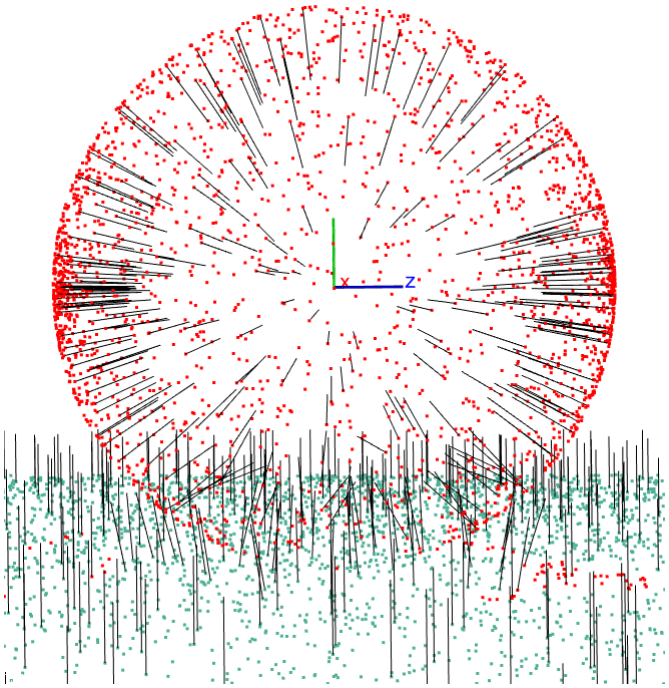
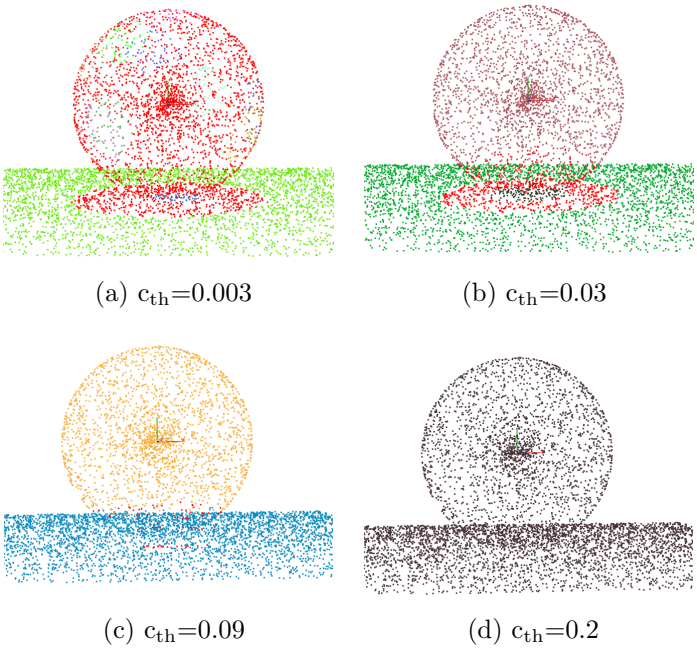


Figure 3.34: Normals visualization for Figure 3.33 - normals in the sphere-plane contact area are noisy

- **Curvature threshold $\mathbf{c_{th}}$**: allows a point to be added as a seed for further points. Points are confronted with seeds in order to be added to the cluster, updating the list of seeds permits the cluster to grow over the initial possible set, which is given by ϑ_{th}. As explained in section 3.1.4 the point curvature is obtained together with the normal, so the curvature threshold shares the

properties explained for the smoothness threshold. Raising c_{th} let the cluster grow among surfaces with higher curvature and its effect is demonstrated in Figure 3.35, using the same scene as before. A low value for this parameter



(a) c_{th}=0.003 (b) c_{th}=0.03

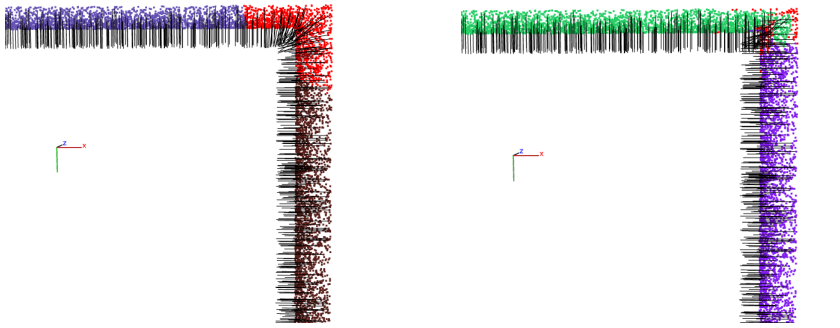
(c) c_{th}=0.09 (d) c_{th}=0.2

Figure 3.35: Curvature threshold effect on a sphere generated with random points - red points can't be assigned to a cluster, other colors correspond each to a different cluster. Other parameters: ϑ_{th}=50 , N_{min}=50 , N_{max}=1000000.

means that a cluster can unlikely grow itself and the algorithm will end up having several sets with a low number of points (see Figure 3.35.a), because the sphere is divided in many colors. The sphere is a surface that has constant curvature, so when c_{th} reaches its value then the shape can be fully identified (see Figure 3.35.b and Figure 3.35.c), excluding areas with noise coming from intersection from different surfaces. Similarly to what happens for the smoothness, a too much high threshold, results in a unique cluster containing the whole initial data set (see Figure 3.35.d).

Unlike the RANSAC paradigm, the Region Growing Clustering allows to tune also the creation of the normals, that are requested as an input in the PCL implementation. This can be considered as an additional degree of freedom in the procedure, although manipulating it does not have immediate, intuitive result and should be done carefully; its effects are displayed in section 3.1.3. It still can be really useful

in simple situations, like splitting two different, incident surfaces. In this case there are two possible strategies: have a large neighbor to calculate the normals (i.e. smooth transition between close normals) and detect minor changes in their orientation to identify the change of surface. Or, at the opposite, use a small neighbor (i.e. possibility of noise on the normals) and search for great orientation changes. In Figure 3.36 is provided an example using two planes, built from random points, forming a 90 degrees edge.



(a) Big Neighbor approach: N_{min}=50, N_{max}=10000, c_{th}=1, $\vartheta_{\mathbf{th}}$**=0.1**, $\mathbf{N_{neigh}}$**=100**

(b) Small Neighbor approach: N_{min}=50, N_{max}=10000, c_{th}=1, $\vartheta_{\mathbf{th}}$**=4**, $\mathbf{N_{neigh}}$**=10**

Figure 3.36: Corner segmented with Region Growing, using two different approaches

Using planes registered with equally spaced points, the two strategies have the same, successful, output, thanks to their lack of imperfection. Using a noisy data set the "small neighbor" approach has an error in the identification, where the surfaces touch, and, in general, the output is more sensible to changes in the parameters. On the other hand the "big neighborhood" test shows a no-detection zone exactly in correspondence of the corner and clearly distinguish the different planes. This approach is more safe with respect to the value of the parameters, as the only major difference is the shrinking or enlarging of the corner zone, so it is chosen as the best.

3.3.3.2 Results

In this section, first, segmentations on the same scenes used for the RANSAC algorithm in section 3.3.2.5 will be performed and, eventually, also tests on real acquisitions are presented. A main difference between the two is that the Region Growing produces many clusters, so it's possible to identify many objects within a

single segmentation. Figure 3.37 give a visual representation of the output, whose quality is better explained in Table 3.26a.

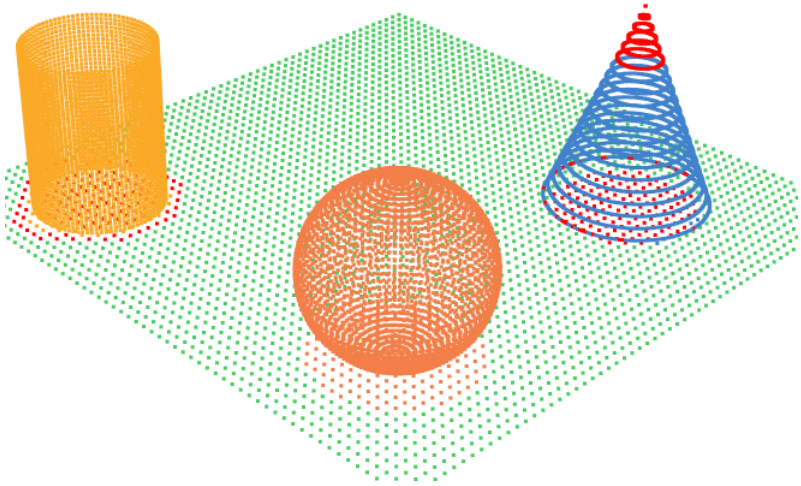


Figure 3.37: Region Growing Segmentation on a computer constructed scene - red points cannot be assigned to any cluster

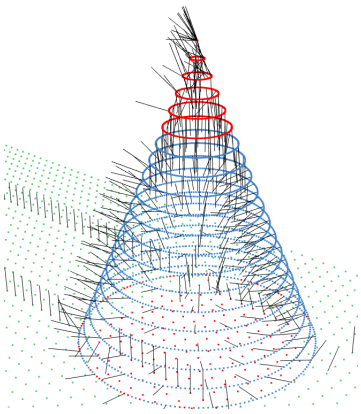


Figure 3.38: Poor normal estimation near the cone's vertex

The percentages of correctly identified points are lower respect to the RANSAC case, for each shape. Analyzing each object separately it has to be noted that, for the plane, most of the points that are missed are the ones inside the cylinder and cone bases, which can't be present in a real acquisition (they would be covered), so this percentage is expected to grow in a real registration. The sphere and the cylinder are well segmented, due to their constant curvature, while the cone is the

Shape	# of original points	# of identified points	% of correctly identified points
Plane	4000	3087	77.17
Sphere	8100	8273	97.86
Cylinder	3600	3654	98.5
Cone	3620	2497	68.75
Not Identified		1409	
SCENE	19320		
Time Elapsed	1119		

(a) Results of the segmentation

c_{th}	ϑ_{th}	$\#_{neigh}$	N_{min}	N_{max}
0.1	20	200	1000	1000000

(b) Parameters set for the segmentation

Table 3.26: Region Growing Segmentation result for a computer constructed scene

worst of all four shapes, this happens due to the poor normal estimation on its vertex, where they change orientation suddenly, as can be seen in Figure 3.38. This problem is related to the construction of the cone, in fact each "horizontal slice" has the same number of points, resulting in a high point density near the vertex; also this issue should be minimized in a real data set.

The Region Growing algorithm can handle robustly noisy data set, differently from the RANSAC, which has to adapt its parameter to the level of noise (whose knowledge is not always predictable) and nevertheless could fail the recognition. Figure 3.39 and Table 3.27a show the result for a segmentation performed on a data set with an appropriate noise and what stands out is that the performances are not far from the ideal case and it even seems to perform slightly better. This small rise in the identified percentages is manly due to a better normal reconstruction in the surface contact areas and on top of the cone.

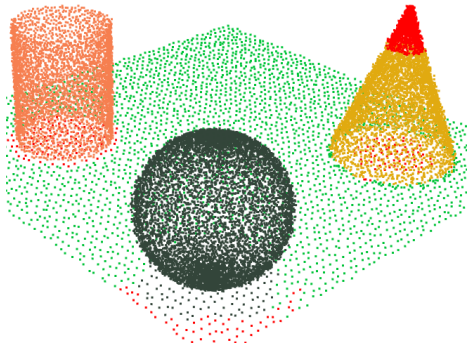


Figure 3.39: Region Growing segmentation on a reconstructed scene with added noise - red points cannot be assigned to any cluster

Shape	# of original points	# of identified points	% of correctly identified points
Plane	3600	3111	86.41
Sphere	8100	8277	97.8
Cylinder	3600	3627	99.25
Cone	5611	4127	73.55
Not Identified		1769	
SCENE	20911		
Time Elapsed	1119		

(a) Results of the segmentation for the noisy scene

c_{th}	ϑ_{th}	$\#_{neigh}$	N_{min}	N_{max}
0.1	20	200	1000	1000000

(b) Parameters set for the segmentation

Table 3.27: Region Growing Segmentation result for a computer constructed scene for the noisy scene

Partial shapes are also detected with reliability in most cases (see Figure 3.40) and in general surfaces whose normals collide have more chance to be easily identified correctly. On the contrary, for example, a partial sphere has normals on the contact area that have similar orientation with the plane, so its recognition is tougher with this procedure. This issue is present not only for recreated scenes, but also in real ones, referring to Figure 3.41.a and Figure 3.41.b. Well-defined

shapes are recognized with good precision (99,67 % and 98.89% of correctly identified points for a sphere and a cylinder respectively), while in Figure 3.41.c a smooth transition between the cone and its frame prevent a fully correct segmentation. The last step is applying the Region Growing Clustering on the acquisition of real size sails to verify if it could be applied in this work. Unlike the RANSAC, its results are absolutely positive when tested with these data sets, in fact it is able to recognize most of the points belonging to the sails, see Table 3.28a. Most important, these clustering allow to extract almost all the interested points, while the major part of the errors come from extra points included, which can be eventually removed via filtering or through an human operator intervention.

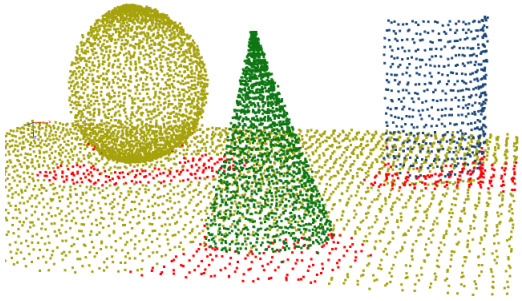
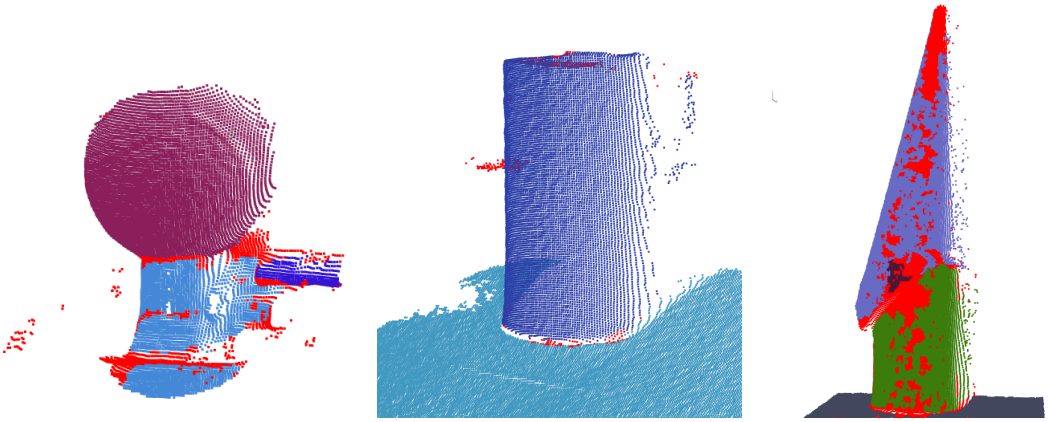
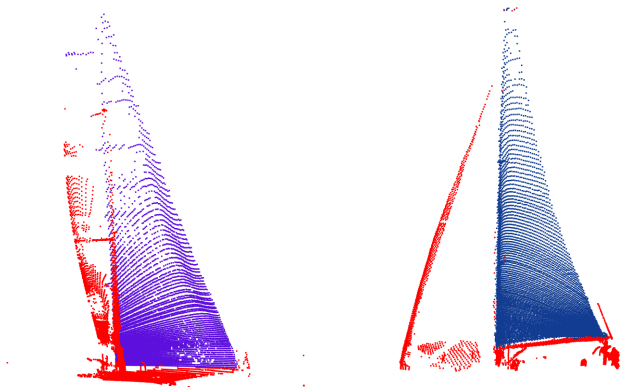


Figure 3.40: Region Growing Segmentation results for partial shapes - Cone correctly identified points: 99.25% - Cylinder correctly identified points: 89.35%



(a) Sphere (b) Cylinder (c) Cone

Figure 3.41: Region Growing Segmentation on real acquisitions



(a) Gennaker (b) Mainsail

Figure 3.42: Region Growing Segmentation with on-field sails acquisitions

	# of original points	% of correctly identified points	Time Elapsed [ms]
Gennaker	8837	97.17	514
Mainsail	7054	89.58	1462

(a) Results

	c_{th}	ϑ_{th}	$\#_{neigh}$	N_{min}	N_{max}
Gennaker	0.01	5	50	6000	100000
Mainsail	0.025	15	20	5000	100000

(b) Parameters

Table 3.28: Region Growing Segmentation performed on on-field sails acquisitions

The presented segmentations, until now, are referred to acquisitions on open air field, it is following showed that the same procedure, adapting the parameters, works also on acquisitions recorded during wind tunnel campaigns. Figure 3.43 displays the segmentation performed on two wind tunnel acquisition with the parameters reported in Table 3.29. The first data set is dedicated to the gennaker acquisition, while the second to the mainsail. It is evident that a unique scan could not acquire both the sails, since the gennaker, in the second acquisition, is partially covered by the mainsail. Always in Figure 3.43.b note that all the shrouds are separated correctly from the actual sail, while in Figure 3.43.a the forestay become confused with the gennaker. This happens because, in order to acquire correctly all the surface of the gennaker, the acquisition device needs to be placed really close to the model of the yacht (in fact the hull is not visible in the gennaker acquisition).

In this case, the high density of the point cloud, lead not only to the acquisition of a more detailed shape, but also unwanted details (as the forestay is) are acquired with better precision and their transition from the sail is smoother (and thus they are harder to eliminate) respect to the mainsail case, where the shape is recorded from a bigger distance. This issue is not directly solvable with the segmentation, because the acquisition device has to be placed in a position where the sail is acquired completely in a unique scan, to guarantee a proper surface reconstruction.

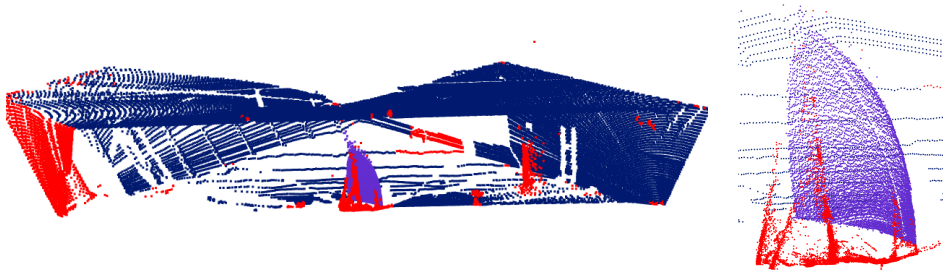
	# of original points	% of correctly identified points	Time Elapsed [ms]
Gennaker	9615	98.8	499
Mainsail	2585	97.1	715

(a) Results

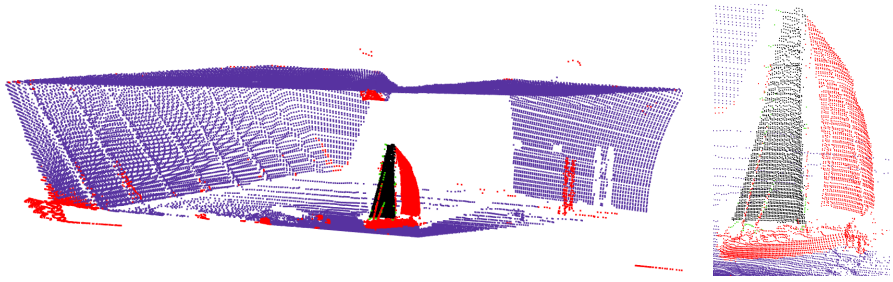
	c_{th}	ϑ_{th}	$\#_{neigh}$	N_{min}	N_{max}
Gennaker	0.9	7	50	5000	100000
Mainsail	0.8	7	50	2200	100000

(b) Parameters

Table 3.29: Wind tunnel segmentation



(a) Wind tunnel gennaker segmentation - full size (left) and zoomed (right)



(b) Wind tunnel mainsail segmentation - full size (left) and zoomed (right)

Figure 3.43: Wind tunnel segmentations

3.3.3.3 Conclusions

Region Growing Clustering proved to be a successful algorithm for extracting a sail from a whole acquisition and more generally, is able to separate different objects from each others. The procedure relies on the local information brought by the point cloud, so the parameters are not influenced by the scale of the measurements. The possibility to set different parameters allows also to have a good control level on the result, so slightly different shapes can be extracted in the best way possible, adapting the input values. Identifying different surfaces can be difficult in case of a smooth transition between them, this could lead to the inclusion of extra point in the segmentation, in this particular work could happen that the spreaders and some sheets are included in the sail cluster. On the other hand the Region Growing has performed great in excluding the forestays from the sails. In the end, a perfect segmentation can't be assured and an eventual human control has to be performed, however this intervention is minimized to clearing a few number of points respect to the initial case.

3.3.4 The Hough Transform

The Hough transform is a technique which allows to detect straight lines, curves or pre-defined shapes within an image, starting from their point-to-point projection onto a parameters space, named "Hough Space". Introduced and patented by Hough in 1962 with [Hough, 1962], it has been expanded by Duda and Hart, [Duda and Hart, 1972], who improved and extended with the detection of further geometrical figures. It's a procedure widely used in digital image recognition, with several employments, as far as faces identification through their generalized shapes as circumferences and ellipses. Developments in the three dimensional space are relatively recent, starting from the 2000's years and particularly interesting is the work of Knopp et al. [Knopp et al., 2010], who achieve shape recognition through the comparison of point cloud features with the features of a pre-defined set. For each data set to be identified they calculate the SURF features [Knopp et al., 2010], which are then compared to the features of a training data set, that is assigned to a shape. This procedure allows to retrieve a wide range of shape classes, not only geometrical, but even far more complex, like animals and people. However the

training features, and so the corresponding set have to be really similar in order to assure a correct identification, meaning that a crouched man, for example, can't be identified as a human figure using a standing man training data set. At the same time objects registered from point of views can encounter difficulties in the features matching. The solution proposed is to create a database of the most possible angle of view for each class that has to be retrieved in the acquisition and perform the procedure with every instance. The creation of such databases is not a demanding work in the case of rigid body classes and nowadays several internet sites that provide them are also available. In this work, however we deal with a particular class, the sail, which not only is a surface more than volume, but is also a deformable surface, which means that the training set should include all the possible and plausible deformation for each point of view. This would lead to a huge database, whose composition is not immediate, because in some sailing trims, the sail shape is not stable and difficult to register. A different approach is used by Salzmann et al. in [Salzmann et al., 2007]: they deformate the triangulated CAD design shape to obtain the set of training shapes. In every case the construction of a database related to the application discussed in this work would exceed the goals of the current thesis, having found an effective, alternative method to obtain the same result.

3.4 Data Resampling

Data resamplings are procedures that upsample or downsample the general, or local, density of a point cloud, using averaging or interpolation of the original dataset. The need for resampling an already acquired geometry comes directly from the generation of the data: they are sampled from a real surface and inevitably prone to noise inaccuracy. This category of techinques is mostly used for its properties of noise filtering and surface smoothing. In cases where the target presents a smooth surface, this feature can easily be lost in the acquisition process, because the presence of noise could generate spikes or holes. Smoothness has to be recreated through these post processing techniques. Their main property, as said, is that the overall density of the filtered cloud is not similar to the original one. Figure 3.44 shows the effects both in the downsampling and in the upsampling case.

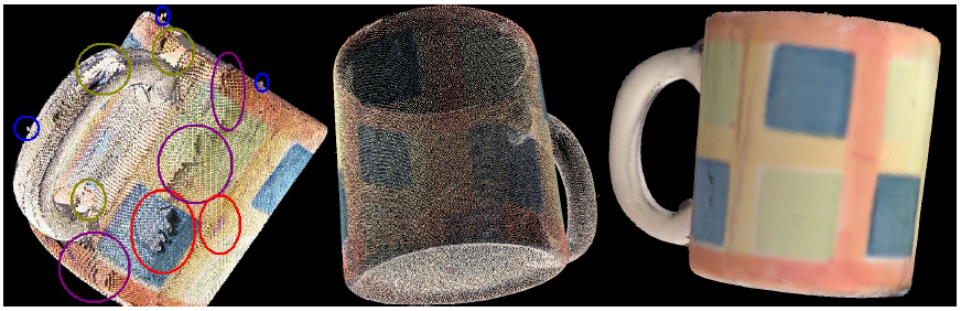


Figure 3.44: Effect of data resampling: original cloud (left) with highlighted acquisition imperfections - downsampled cloud (center) - upsampled cloud (right)

Performing a data resampling can be useful in removing local imperfections. These defects might occur for different reasons such as variable cloud density (see section 2.4.3), overlapping areas due to error in registration of pairs of clouds (see section 3.2.3), surface properties that do not allow for distance measurements, causing holes in the cloud and sharp edges that result in mixed pixel problems (see section 2.4.3). Some of these irregularities can not be easily deleted through traditional filtering, such as radius or statistical outlier removal but have to be taken into account since they might compromise the data post-processing.

Standard resampling methods made their way into robotics applications from the computer graphics research community [Levin, 2004], and are usually formulated as Moving Least Squares (MLS) solutions [Alexa et al., 2003]. MLS methods provide an interpolating surface for a given set of points P by fitting higher order

bivariate polynomials to each point neighborhood locally. In contrast to other interpolation or resampling techniques, MLS has the advantage that the resultant fitted surface passes through the original data points. The data resampling effects are not always immediately visible on the point cloud, as visible in Figure 3.45 that represents the acquisition of a planar surface. Pre and post filtering clouds do not seem to be changed, but the analysis of the point normals reveals the differences: acquisition noise is reduced leading to curvature values constantly around zero, as expected for this kind of surface. This is the reason why data resampling is not an elaboration that stands by itself, but it is an important step that is performed before applying others operations on the cloud, especially triangulation or surface reconstruction.

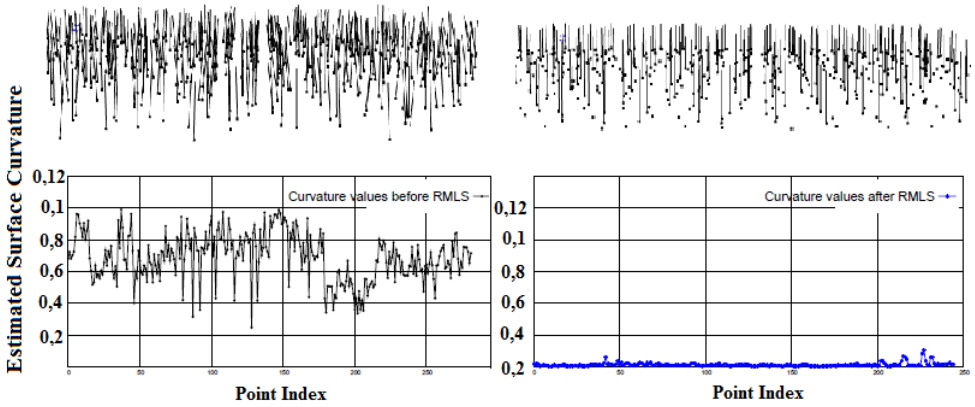


Figure 3.45: Effect of data resampling - Normals and curvature pre-filtering (left) and post-filtering (right)

The approach beyond data resampling is motivated by differential geometry, which assumes that the surface approximation can be locally expressed as a function, and aims at minimizing its geometric error. This is done by locally approximating the surface with polynomials using MLS. Starting from a data set of points $P = \{p_i\}$, a smooth surface S_p is defined, representing the input points. The points P are reduced, defining S_p with a reduced set $R = \{r_i\}$ and introducing another MLS surface S_r which approximates S_p. The general paradigm is illustrated in 2D in Figure 3.46: Points $p_i \in P$ are depicted in purple and define a curve S_p, also in purple (3.46.a). S_p is then re-sampled with red points $r_i \in S_p$(3.46.b). This typically lighter point set is called the "representation points" and defines the red curve S_r which approximates S_p. The representation points set is the output of the

procedure and is the point cloud shown in Figure 3.44, center and right], upsampling or downsampling is determined by the curvilinear sampling distance of the R set.

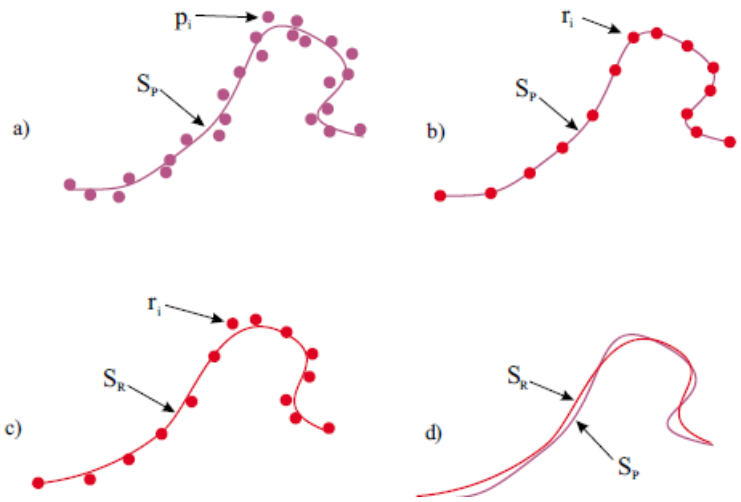


Figure 3.46: Resampling paradigm, in a two dimensional space - The original points p_i are approximated through a polynomial line S_p(both in purple), see part (a). S_p is then resampled with a fixed point-to-point curvilinear distance, generating r_i, see part (b). The polynomial approximation of r_i is S_r, which approximates also S_p, see (c),(d).

The key part of the procedure is creating the approximating surface (lines in 2D) for each point, starting from the data set P or R, and can be divided into two steps: the identification of a reference domain and the effective computation of the minimizing surface.

The local reference domain H of a point r is constructed in order to minimize a local, weighted sum of square distances of the points p_i to the plane, where p_i are the neighbors points in the volume considered. It is done similarly to the creation of a best-fitting plane of the set, although the points are not weighted in the same way, but weights attached to p_i are defined as the function of the distance of p_i to the projection of r on the plane H, rather than the distance to r (see Figure 3.47). Assuming q is the projection of r onto H, then H is found by locally minimizing:

$$\sum_{i=1}^{N} (< n, p_i > -D)^2 \, \theta \left(\|p_i - q\| \right) \tag{3.38}$$

where θ is a smooth, monotone decreasing function, which is positive on the whole

space. A local reference domain is set up and it is then given by an orthonormal coordinate system on H so that q is the origin of this system.

In the second step the reference domain H for r is used to compute a local bivariate polynomial approximation of the surface, in a neighborhood of r. Let q_i be the projection of p_i onto H, and f_i the height of p_i over H (see Figure 3.47), i.e $f_i = n \, \mathring{u} \, (p_i\text{-}q)$. The polynomial approximation g is created, computing the coefficients that minimize the weighted least squares error:

$$\sum_{i=1}^{N} \left(g\left(x_i, y_i\right) - f_i\right)^2 \theta\left(\|p_i - q\|\right) \tag{3.39}$$

The projection P of the point r onto S_p, which is the returned output of the whole procedure, is found evaluating the polynomial value at the origin of the axes, i.e. $P(r) = q + g(0,0)n$.

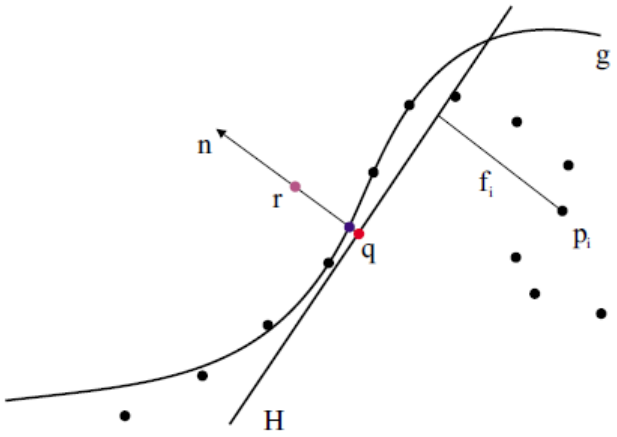


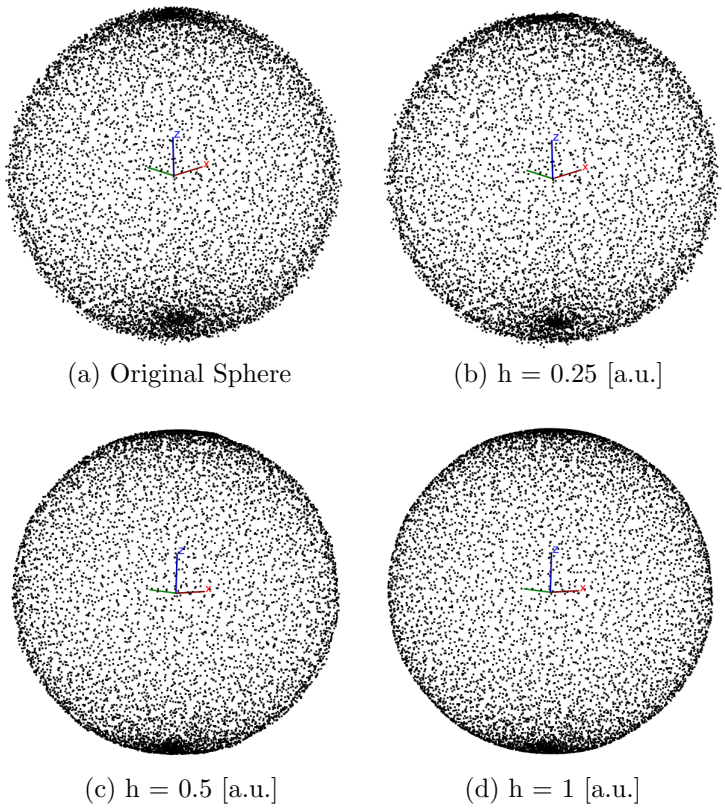
Figure 3.47: The MLS Projection Procedure - First, a local reference domain H for the purple point r is generated. The projection of r onto H defines its origin q (the red point). Then, a local polynomial approximation g to the heights f_i of points p_i over H is computed. In both cases, the weight for each of the p_i is a function of the distance to q (the red point). The projection of r onto g (the blue point) is the result of the MLS projection procedure.

In the procedure, the only user definable term is the radial weight function θ, which is the parameter that influences the density of the output point cloud, because it determines the volume (or area in a two dimensional environment) considered for calculating the neighborhood of each point. The most widely used weight function was first proposed by [Levin, 2004], as a simple Gaussian function:

$$\theta(d) = e^{-\frac{d^2}{h^2}} \tag{3.40}$$

where h is the parameter reflecting the neighborhood size.

The incidence of h is presented in Figure 3.48 too, where smoothing is applied on a sphere, generated with a percent level of noise comparable to the TOF sensor as described in section 2.4.3, using different h values.



(a) Original Sphere (b) h = 0.25 [a.u.]

(c) h = 0.5 [a.u.] (d) h = 1 [a.u.]

Figure 3.48: Effect of different values of h on a sphere (r = 4 [a.u.]) with noise - Noise amplitude = 0.12 [a.u.]

The noise reduction effect as a function of h can be evaluated also in quantitative terms computing the sum of the quadratic distances between each resampled point and the surface of the noise-free sphere (see Figure 3.49).

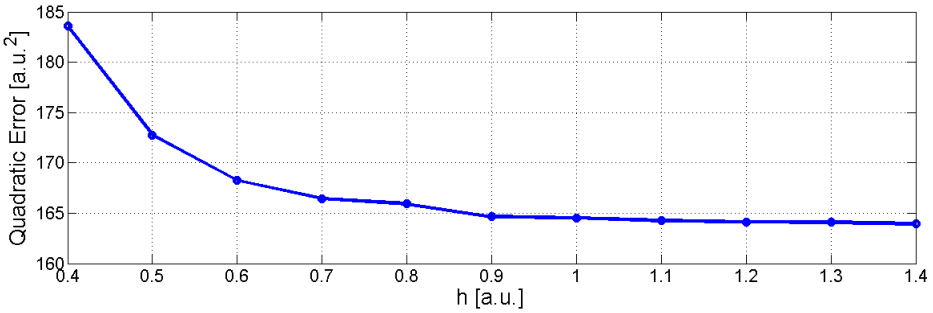
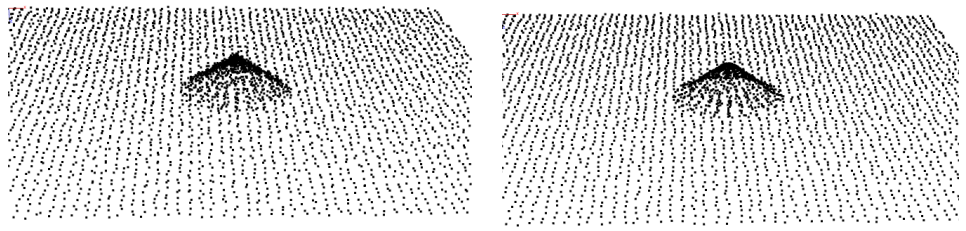


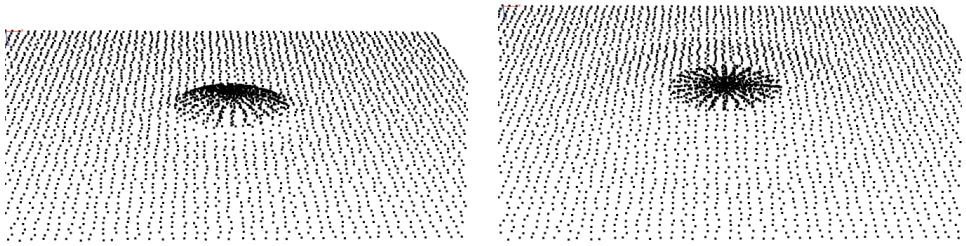
Figure 3.49: Global quadratic error as a function of h

Values of h smaller than 0.4 could not be included because resulted in a subsampling of the data set, hence the impossibility to compare it with the model through the quadratic error function. It is however evident that the error decreases rapidly in the first part of the plot, but increasing h over 0.9 has almost no effect.



(a) Original feature

(b) h = 1 [a.u.]



(c) h = 5 [a.u.]

(d) h = 10 [a.u.]

Figure 3.50: Elimination of sharp features - The original cone has radius equal to 3 [a.u.] and height equal to 2 [a.u.]

This limit value could not be fixed but has to be adapted considering the data set density and noise level. In fact, setting a high value for h is not always a good choice, because it could smooth out surface details (smaller than h) that might be of interest. More specifically, a small values for h cause the Gaussian weight function to decay faster, thus the approximation is more local. Conversely, large values for h

result in a more global approximation, smoothing out sharp elements of the surface, as demonstrated in Figure 3.50. In Figure 3.50b the neighborhood is smaller than the feature dimensions, so it is only smoothed, but not deleted. As the value of h increases the cone is progressively "absorbed" into the planar surface altering the data acquired. This issue is particularly critical for the application considered in the present work, as the final purpose is to retrieve a flying sail shape which could be used for CFD or geometrical measurements. On the other hand, resampling of the data is required to remove the acquisition noise and achieve better results.

The data resampling affects not only the point spatial position, but also its normal and curvature. Some tests were performed to analyze this aspects. A point cloud was generated sampling a CAD design sail surface, which was used as reference. Normals and curvatures for this model cloud were computed. Then, a new cloud was created applying a 0.27% noise on each point of the reference cloud (worst case scenario from section 2.4.3). A second cloud came from the resampling of the noisy data set, with the minimum neighborhood radius which avoid subsampling.

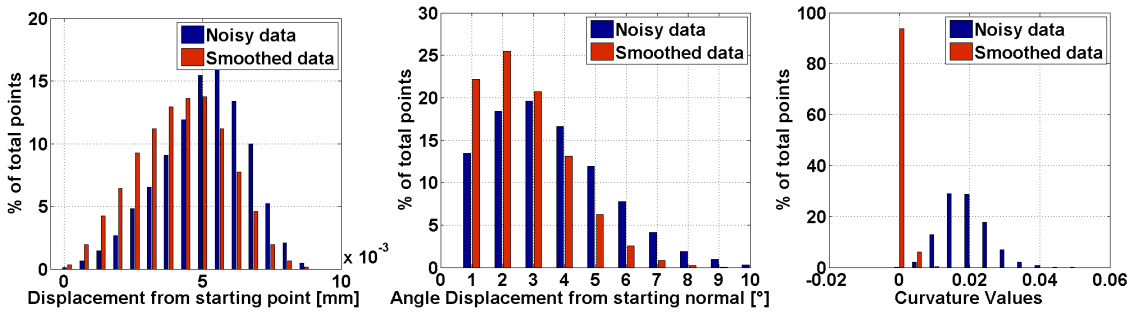
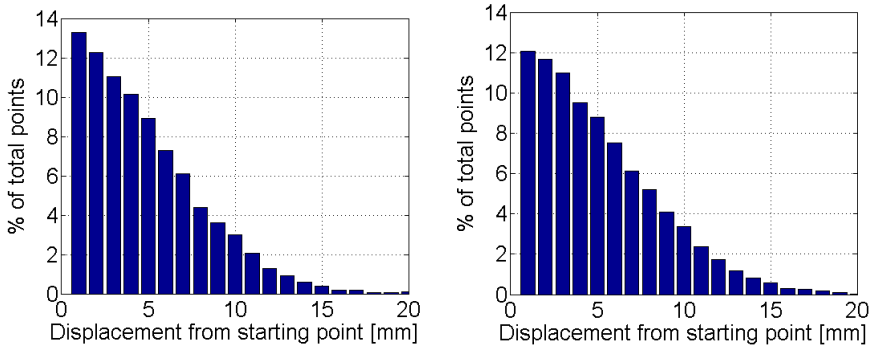


Figure 3.51: Data resampling effect: a noisy data set is smoothed and then confronted with a model set

These two sails are compared in terms of point spatial displacements, normal deviations and curvature differences from the model sail (computed for each point in the cloud). As visible from all the histograms in Figure 3.51, the smoothing moves the distributions towards smaller values, which means that the points resampled are closer, and have closer normal and curvature, to the model points. This is for sure a positive effect, however, this distributions take in account only local information from a single point and do not consider changes in the neighborhood. Global effects on changes of the overall shape of the sail are much more difficult to assess, especially for real acquisition, when the comparison with a model sail

is not possible. To investigate the effects of data resampling applied on the sail scans acquired during wind tunnel tests, resamplings with different neighborhood value have been performed. Each smoothed cloud is then compared to the original one in terms of point displacement. Figure 3.52 shows the distribution of these displacements.



(a) h=60 mm (b) h=80 mm

(c) h=100 mm (d) h=200 mm

Figure 3.52: Histograms of the distributions of the displacement of the sail points after resampling

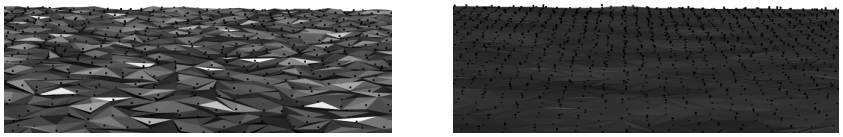
h=60 mm is the minimum value to avoid subsampling the data set and increasing it moves the points far away from their original positions, as highlighted also in Table 3.30. Points are divided as function of their displacement from the starting point and three main intervals can be identified. A point is labeled as "good point" if it is relocated in a position whose distance from the original one is smaller than the measurement uncertainty (it cloud be considered affected by noise). From Table 2.16, a point falls in this case if it is moved less than four millimeters. With greater displacements two situations could happen: the point is moved but still belongs to the actual sail surface or it is moved so far that it could not belong

anymore. Again, the limit value is identified from the metrological qualification of the laser scanner. From Table 2.15 the systematic error at two meters (height of the sail) from the acquisition device is equal to five millimeters. To reach a confidence value, over whom the point is not on the sail, also the statistical error is added. The threshold value is then placed at nine millimeters. Points moving over the threshold are considered "bad points", all remaining samples (displacement between four and nine millimeters) are considered "fair points". A fair point represent a point which is moved, but within the limit of alteration of the shape.

h [mm] =	60	80	100	200	400	1000
good points %	46.7	44.2	43.7	42.8	39.3	17.2
fair points %	44.2	44.3	42.5	43	43.4	26.2
bad points %	9.1	11.5	13.8	14.2	17.3	56.6

Table 3.30: Percentages of small (good) or great (bad) displacements in the resampled data set

"Bad" points are not necessarily undesired, but there is the possibility that they do not belong to the surface acquired. They are the result of the resampling procedure, which, generally, generates points which belongs to a more smoothed surface, as visible from Figure 3.53. A data set representing a noisy plane has been created and then an adequate smoothing has been applied. The surfaces are generated with the Greedy Projection algorithm, explained in section 3.5.1 , with the same parameters.

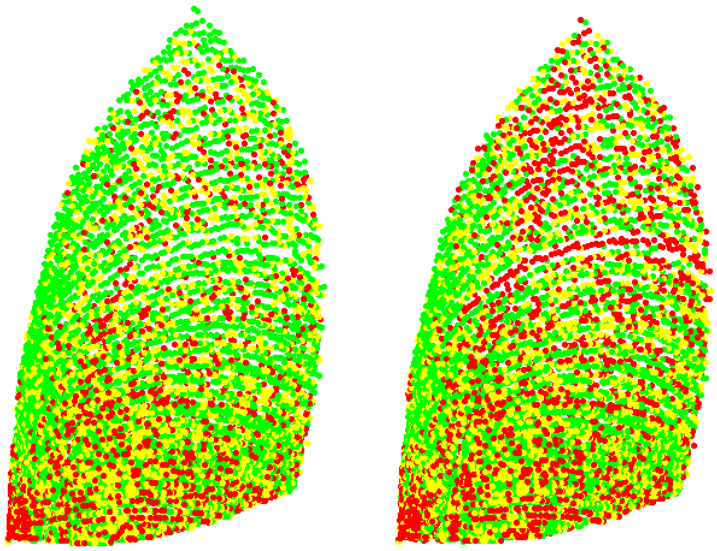


(a) Pre-resampling surface (b) Post-resampling surface

Figure 3.53: Data resampling effect on surface generation from points of a noisy plane

Point displacement due to the resampling process is more severe for points at the edge rather than inside the acquired cloud. The reason could be that these points present an "asymmetrical" neighborhood, because the local neighborhood is created as a sphere centered in the considered point. It is obvious that points on the

border (or near) of the sail have neighbors condensed in a unique direction, rather than all around them. Their movement affects the measures of the edges of the sail, which is one of the parameters which are used to check for a correct acquisition; if the measured edges are not close to the design values, than the acquisition is considered to be faulty. In order to evaluate the position of the points that move the most after resampling, the data set is plotted and a colormap, which replicate the displacement of each point, is applied. The result is visible in Figure 3.54, for two different resamplings,in order to visualize the effective differences from one operation to the other.



(a) h=60 mm (b) h=200 mm

Figure 3.54: Data sets with displacement colormap: green points are moved less, red points are greatly moved

In the left figure, red points are more scattered, and concentrate only in the bottom left corner, while in the other picture, there are different areas of greater movements, obviously. The edges of the sails seem to be almost untouched in the first smoothing shown, which is what is required. The lower part of the shape is the one that is more modified, this happens because it is also where the local density is higher, so each point has more neighbors and the interpolation is more marked. The presence of more points however, guarantees that the interpolation is closer to the real shape.

A critical analysis on the points and their displacements has to be performed for every shape acquired, since data resampling can't be applied in a complete

automatic way. A good result depend not only by the parameter applied , but also from the input data set. If the input is not well-filtered, in other words the outliers have not been removed, then the resampling interpolate also these unwanted points, creating big displacements. In the cloud shown in Figure 3.54, the bottom left corner is affected by the presence of some points not belonging to the sails, but probably to the jib sheet, as visible in Figure 3.55. These points influence their neighbors, generating bigger movements than needed. Note that the high density of the data set in this area reduces the influence of bad points, since the general shape is still maintained, due to the presence of good points.

After the analysis of several point clouds, $h = 80$ [mm] as been selected as standard value to perform the smoothing. The resulting sails has been approved by a sail maker, which confirmed that with the chosen parameter value, the overall shape does not change .

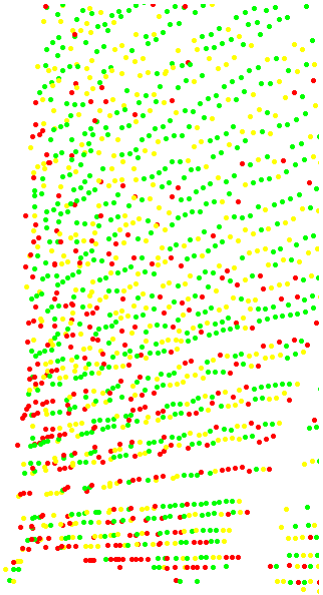


Figure 3.55: Sail corner detail after resampling (h=60 mm), points not belonging to the sail generates local big displacements on the sail samples

Similar observations can be made on acquisitions of real sails. Of course the limit for "good" and "bad" points have to be changed, accordingly to the different size of the sail. In the following test a mainsail is considered; its maximum height is seven meters. Good points are always defined as samples whose displacement is smaller than the noise threshold (four millimeters), while the limit for bad points is moved to 14 mm.

h [mm] =	175	225	275	350	500	750	1000
good points %	62.3	61.3	61.3	59	54.1	42.5	33.1
fair points %	37.7	38.7	38.68	40.9	45.3	50.5	48.5
bad points %	0	0	0.02	0.1	0.6	7	18.4

Table 3.31: Percentages of small (good) or great (bad) displacements in the resampled data set

Obviously also the neighbors radius has to be adapted to the point cloud, because greater acquisition distance lead to sparser samples and to the need of a larger neighborhood, to avoid subsampling. Moreover, during acquisition of real sails, the main problem encountered has been the loss of data in the higher part, as explained in section 2.4.3. This issue cause holes in the sampling, and so the neighborhood has to be even more enlarged to deal with them.

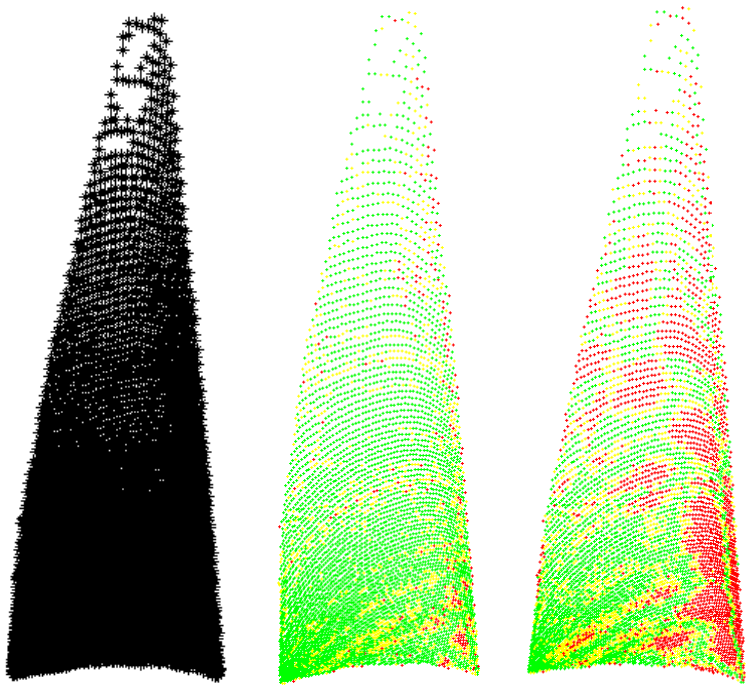


Figure 3.56: Mainsail resampling: the original acquisition (left) is 12 m high, resampling with neighborhood radius equal to 350 mm (center) and 750 mm (right). Red points are bad points, yellow points are fair points and green are good points.

The starting value for avoiding subsampling, in this case, is 175 mm. Several resampling has been considered, whose results are showed in Table 3.31. Here the situation is much different, because the influence of noise decrease with the

acquisition distance (see Table 2.15) and so a larger neighbor can be used more with more confidence. The main restriction remains on the correct length of the sail border, which is influenced by the position of bad points. As visible from Figure 3.56, bad points concentrate on the luff, which could affect the measure of this side. Here the holed previously mentioned can be seen on the top part of the mainsail, the procedure can't fill them. The selection of a standard value for the smoothing parameter has to adapt to the dimension of sail. After several tests $h = 500$ [mm] has been chosen, but in most cases it depends mostly on the quality of the single acquisition.

3.5 Surface Modeling

The problem of surface reconstruction from unorganized point clouds has been, and continues to be, an important topic of research. It allows fitting of scanned data, filling of surface holes, and remeshing of existing models. The problem can be states as follows: given a set of points $\mathbf{P}$, which are sampled from a surface in $\mathbf{R^3}$, construct a surface $\mathbf{S}$, so that the points of $\mathbf{P}$ lie on $\mathbf{S}$. From this definition follows that $\mathbf{S}$ is an interpolating surface, with respect to $\mathbf{P}$. [Gopi and Krishnan, 2002] classify reconstructing methods in two categories: the explicit methods attempt to reconstruct the topology and the geometry of the original surface directly from the given points (that is interpolating them), whilst the implicit methods approximate the point cloud (that is they fit the points).

There is a wide range of applications for which surface reconstruction is important, from medical imagery, cinema special effects, computer vision, reverse engineering, onto virtual reality. Surface representations are a natural choice because of their applicability in rendering applications with a surface-based visualization. Moreover, in the past years, this field has received increasing attention due to the ever broadening range of geometric 3D sensors, especially the Microsoft Kinect. It is important to go through the concept of visualization and to understand what it actually means. Visualization in its broadest terms represents any technique for creating images to represent abstract data; lots of different definitions may be given, depending also on the context in which that is used. This thesis deals with the so-called scientific visualization. It can be intended as any technique involving the transformation of data into visual information, using a well understood, reproducible process [Carlson, 2003]. Scientific visualization identifies the field in computer science that encompasses user interface, data representation and processing algorithms, visual representation and other sensory presentation such as sound or touch [DeFanti and Brown, 1991]. It is worth noticing the fact that this process does not serve just to bring existing facts to a visual form, but it can be seen like a method useful to organize and improve our knowledge about the reality. This field of research takes advantage of the natural abilities of the human vision system, that is our most powerful sensory system, on which humans rely in almost everything we do. With the introduction of computers and the ability to generate enormous quantities of data, visualization has been offering the technology to make the best use of our highly developed visual senses. Certainly other technologies such as statisti-

cal analysis, artificial intelligence, mathematical filtering and sampling theory will play a role in large-scale data processing. However, because visualization directly engages the vision system and human brain, it remains an unequaled technology for understanding and communicating data. The goal is to create the model of an object which best fit the reality. Polygonal meshes are the commonly accepted graphic representation, with the widest support from existing software and hardware. A two dimensional mesh is defined as a collection of vertices, edges and faces (see Figure 3.57) that define the shape of an object. Three dimensional meshes have also cells. The most common choice are triangular meshes, which allow to express the topological properties of the surface with the best trade-off between numerical robustness, algorithmic simplicity and efficient display. The underlying, intuitive, reason is that a surface can be locally approximated as a plane, which, in turn, is identified by a minimum of three points. The use of triangular meshes is nowadays a standard in computer graphics applications.

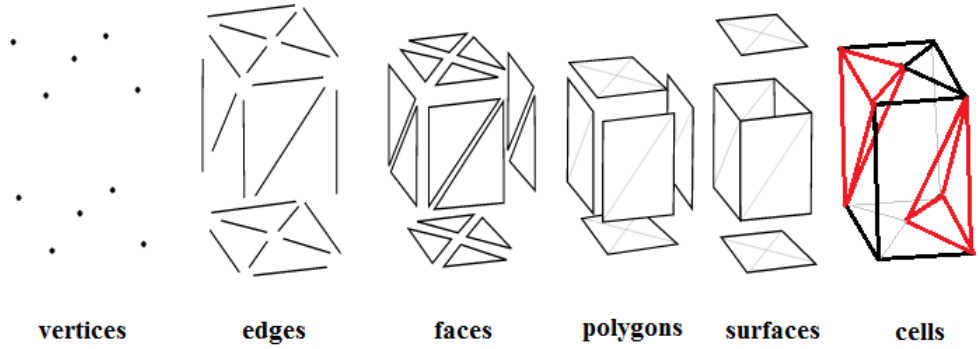


Figure 3.57: Elements of a triangular mesh

In [Mücke, 1993] Mücke stated that, for a set of points $\mathbf{T}$, a surface or volume representation is possible through a k-simplex, denoted as σ_T, where k=$|\mathbf{T}|$-1. For a set with a single point, the only representation possible is a 0-simplex, which is the point itself, or vertex; a two-point set can be divided in two vertices and a 1-simplex, which is an edge. A three points set forms a 2-simplex, or triangle and a four points set forms forms a 3-simplex, which is a tetrahedron, in addition to the ones said before. The tetrahedron is chosen as the smallest unit to describe a three-dimensional geometry, while k-simplices with k>3, which represent four-dimensional units, are not used, due to the significance lack to human visualization. Mücke also stated that a proper reconstruction is achieved when the set of all the simplicies, $\mathbf{C}$, satisfy the following properties:

1. if $\sigma_T \in \mathbf{C}$ then $\sigma_{T'} \in \mathbf{C}$ for every $T' \subseteq T$. In other words, for every simplex σ_T, **C** contains all the faces of σ_T as well.

2. if $\sigma_T, \sigma_{T'} \in \mathbf{C}$ then either $\sigma_T \bigcap \sigma_{T'} = 0$, or $\sigma_T \bigcap \sigma_{T'} = \sigma_{T \bigcap T'} = \text{conv}(T \bigcap T')$. In other words, the intersection of any two simplicies in **C** is either empty or a face of both.

This properties define a set of simplicies as a simplicial complex; as visible from Figure 3.58 non-simplicial complexes are unable to represent correctly a surface.

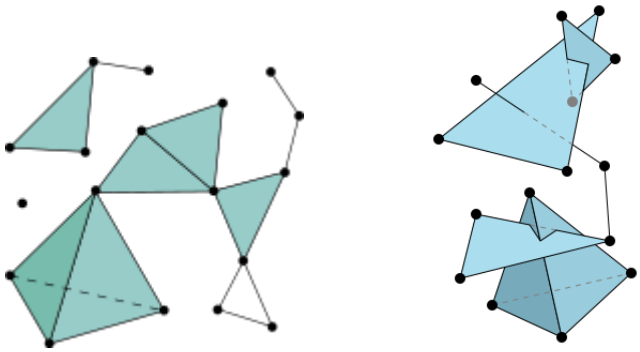


Figure 3.58: Simplicial complexes (left) and non-simplicial complexes (right)

A simplicial complex **C** is called a (geometric) triangulation of a of a set of points **S,** if all vertices of **C** are points of **S** and $|\mathbf{C}|$=conv(**S**). In other words, in addition to the condition of the simplicial complex, the set of vertices of the simplicies have to coincide with **S**. The challenge for surface reconstruction algorithms is to find methods which deal with a wide variety of shape, whose output is, before all, a simplicial complex, and eventually is a good representation of the object considered. The geometric notion of "shape" has no associated formal meaning, in contrast to other geometric notions, such as diameter, volume, convex hull, etc. From a mathematical point of view, a surface in the Euclidean three-dimensional space is defined as a two-dimensional manifold that is compact, connected and contains information about face orientation. In other words, we might say that a surface is a "continuous" subset of points in $\mathbf{R^3}$ which is locally two dimensional. For a surface, a boundary set of points can be defined as the set of points that can be approached both from inside the surface and from the outside. A surface may have a border, when the boundary set is not empty, or it may be closed, when the boundary is empty. Many attempts at reaching a definition of shape has been proposed [Edelsbrunner and Mücke, 1994], [Akkiraju et al., 1995], [Mücke, 1993];

in the current work a shape generically refers to a simplicial complex in the three-dimensional space. Moreover, the interest is focused only on reconstructing specific shapes (mainsails, jibs, gennakers), rather than having an algorithm which can deal with different cases. Several methods are tested on synthetic point sets and then on real acquisitions. A good algorithm has to deal correctly with noisy, unorganized point clouds, which are the output of the acquisition procedure exposed in section 2.3.1 and has to produce non-closed, convex surfaces, since the reconstruction is applied after a successful sail segmentation and, eventually, a smoothing step.

3.5.1 Greedy Projection Algorithm

The Greedy Projection Algorithm, presented by Rusu et al. in [Marton et al., 2009], starts from large, noisy and possible unorganized data sets, recreating the underlying surface as a triangulation of the data set, using an incremental method. It is based on the surface growing principle [Mencl and Muller, 1997], following a greedy type approach. This means that the procedure starts creating a triangle and then keeps on adding new triangles until all points in the cloud have been considered, or no more valid triangles can be joined to the existing mesh. It is notable that once a triangle has been written, then it can not be later changed or deleted. The algorithm works with a three dimensional input, but its output is composed of two dimensional triangular simplicies, which form at least one surface, when possible. The creation of the mesh follows Algorithm 3.6 steps.

Algorithm 3.6 Greedy Projection Algorithm

For each point $p \in P$:

1. Nearest Neighbors Search: a k-neighborhood is selected by searching for the point nearest k neighbors in a sphere with radius $r = \mu \cdot d_0$($d_0$is the distance from the point p to its closest neighbour and μ is a user specified constant).
2. Neighborhood Projection: the neighborhood is projected on a plane that is approximately tangential to the surface formed by the neighbors, which are then ordered around p.
3. Pruning: the neighbours are pruned by distance and visibility criterions.
4. Triangulation: creation of triangles. Each formed triangle has p and two consecutive neighbours as vertex. Possible additional controls are made on the angles between the edges and on the edge length.

Step 2 is the reason why the output is a surface: even though the input are three dimensional points, the actual triangulation is performed after a projection on a local plane. The creation of the neighborhood is computed with the help of a k-d tree structure (see section 3.1.7) to speed up the computation time. Even though the algorithm requires as input not only the spatial coordinates but also the normal information for each point, the tangential plane is found performing an SVD procedure (see section 3.2.1) on a set of near points, instead of simply meaning all the normals. This last option would make sense only if the underlying surface is supposed to be smooth. This hypothesis is not accepted, granting greater generality to the algorithm. Note that the set of points on which the SVD is performed could be different from neighborhood of p, but a different set. Usually it is larger, to reduce the impact of noise on the creation of the plane. After the projection on the tangential plane, points are ordered; this later simplify the creation of the triangles, which are formed following the point order. To achieve a locally coherent orientation a new local coordinate system is defined with the reference point as the origin and the plane projection of the previous step serves as the xy plane. Ordering around the reference point is based on the angle θ between the x-axis of the local coordinate system and the vector from origin to the projected candidate point. Pruning of the points is made with two separate methods, the first, simpler, is the distance criterion, which is applied at the first step of the algorithm, since the neighborhood of each point is defined as the set of points inside a sphere. The remaining points are discarded. Another trimming is eventually performed through a visibility criterion: it removes all the points that do not have a "direct vision" to the reference point, as explained in Figure 3.59.

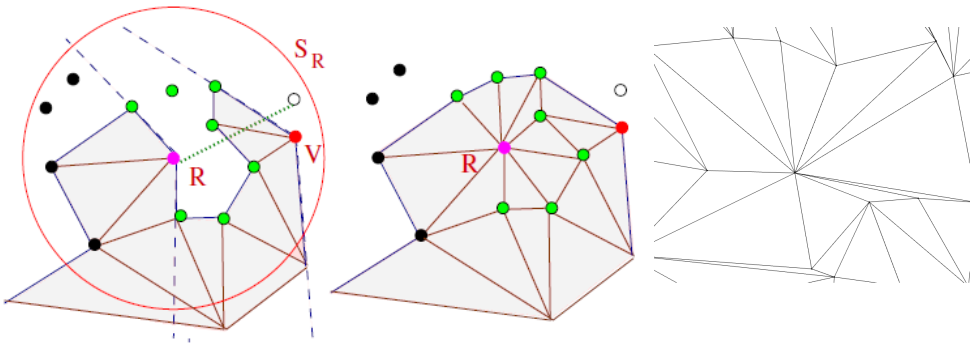


Figure 3.59: Visibility Criterion: before pruning (left), resulting triangles (middle) and visualization of a real triangulation

The visibility criterion is applied on the neighborhood S_R of the reference point R, when part of the mesh is already triangulated. All the considered points could belong, or not, to one or many triangles, but green points have "direct vision" of the reference point, because the segment connecting them does not cross previously formed edges. These are the samples that remain after the pruning and that generate new triangles. The vertices of each new triangle are the reference point and two consecutive neighbor points. When a triangle is generated, before saving it, a check is performed against an angle and an edge thresholds; four conditions has to be verified to confirm a triangle as part of the final mesh:

- Minimum Angle Threshold: the angles between each edge has to be greater than a user specified value.
- Maximum Angle Threshold: the angles between each edge has to be smaller than a user specified value.
- Maximum Surface Angle: the deviation between the normals of the vertex of the triangles has to be smaller than a user specified value.
- Maximum Edge Length: each edge of the triangle has to be smaller than a user specified value.

The first two conditions are introduced in order to eliminate small triangles, which could be the result of noisy data, and big triangles, which could be created due to the greedy approach of the procedure. Small triangles rise the numerical complexity of the mesh, without adding significant information, while on the other hand big triangles could lead to a bad approximation of the surface to be reconstructed. The author identified as good triangles the shapes with angles included between 10 and 120 degrees. The last condition is meant to deal with the cases where there are sharp edges or corners and where two sides of a surface run very close to each other. To achieve a correct reconstruction, points are not connected to the current point if their normals deviate more than the specified angle. Rusu et al. specified that a 45 degrees threshold works on most data sets. This parameter does not have influence for the current application, because the input point cloud represent a smooth surface. However, for a complete analysis of the algorithm, it is however tested first on synthetic shapes, on model point clouds and finally on real acquisitions. The considered shapes start from a plane and gradually get closer to

the sail shape, using a cylinder cut, showing only $\frac{1}{6}$ of the original shape, and a model sail. Evaluating the mesh output is not an easy task, because of its intrinsic visualization nature it is much easier to judge the quality of a mesh for human beings rather than for a computer. In this work, one of the simpler criterion for admitting a mesh as acceptable, is that it does not have holes, since the goal is to reconstruct a continuous surface. Several check can be made on the geometric composition of the triangles to evaluate the surface mesh quality [Frey and Borouchaki, 1999]. It is common practice to avoid building "slim" triangles, which are shapes with the presence of really small angles (less than 10 degrees). Additionally a mesh can be defined as regular, meaning that is composed of triangles similar one to the other, see Figure 3.60 for explanation.

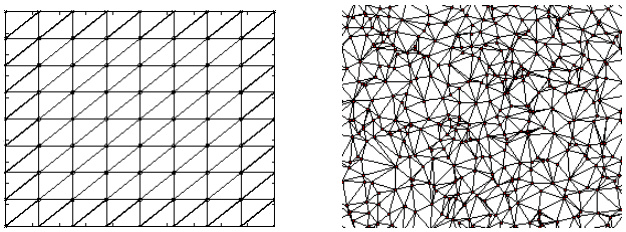
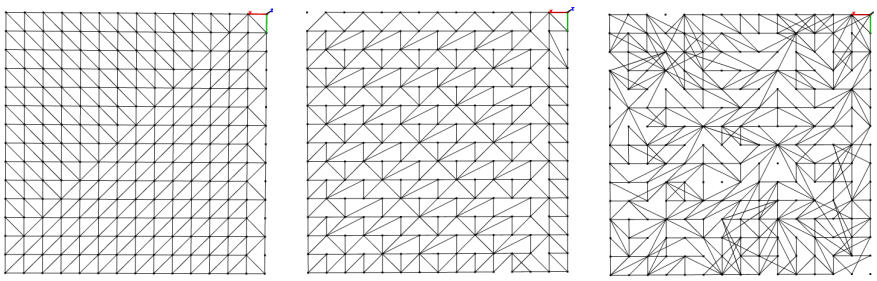


Figure 3.60: Regular Mesh (left) vs. Irregular Mesh (right)

The triangulation created are compared in terms of correctness (lack of holes), angles between the edges and edge length ratio percentages (a regular triangulation have similar edge length, so the major percentage of edge length ratio should be centered around the value one). In the first test performed, a plane created from a 15x15 grid of point, without any noise, the parameters are relaxed, allowing the formation of any type of triangle. The parameter μ is set equal to 1.5; this means that the neighborhood radius is bigger the minimum distance from a point to its closest points. The choice is adequate because of the grid structure of the cloud, with this size each point is evaluated on the projection plane with at least three points (the corner samples). The result of the triangulation is displayed in Figure 3.61.a.

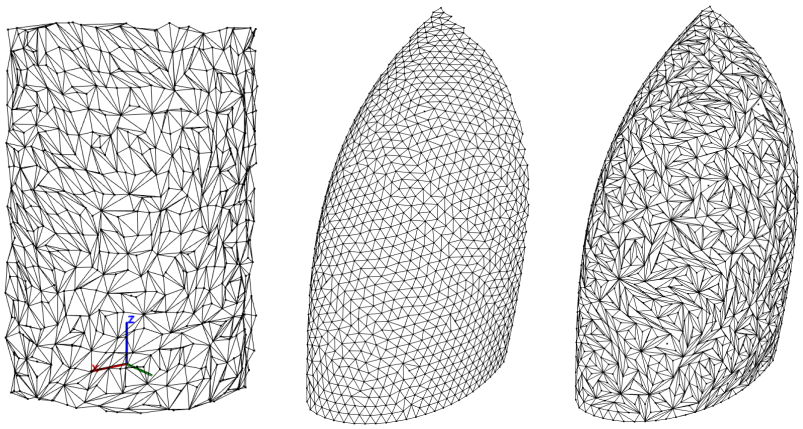
Having a bigger neighborhood for each point does not translate in a more regular mesh. In fact the comparison between Figure 3.61.a, 3.61.b and 3.61.c displays that raising the value of μ creates worse triangulation, this is due to the greedy nature of the algorithms. The first point is connected to every point possible in its neighborhood, creating the first triangles, then other point are added progressively. If the first triangles are large then could happen that following points, especially



(a) Plane triangulation - $\mu = 1.5$ (b) Plane triangulation - $\mu = 2$ (c) Plane triangulation - $\mu = 3$

Figure 3.61: Greedy Projection: plane test results

late points, do not have correspondence to create their triangles. This behavior can be limited through the maximum edge length parameter, however it is better to adapt the neighborhood size to the cloud density, if this information is known. In the next test a real three-dimensional shape is used, which is a cylinder, created via software. In order to make the data set more representative of a real acquisition, the cylinder is cut, showing only 60 degrees of the complete object and noise is added. In this case the radius of the neighbors has to be increased up to $\mu = 6$, to avoid the presence of holes on the reconstruction, while the angle thresholds remains relaxed. Figure 3.61.b shows the resulting triangulation.



(a) Cylinder triangulation (b) Sail model triangulation 1 (c) Sail model triangulation 2

Figure 3.62: Greedy Projection: cylinder and sail model triangulations

After the cylinder, a sail model is considered. This shape is formed from a CAD model of the sail, given by the manufacturer, with a regular sampling. It is

the most ideal acquisition for a sail, since it is uniformly sampled and without any noise. A first triangulation is performed, with a minimum neighborhood size that avoids holes, as visible in Figure 3.62.b. With $\mu = 1.75$, the triangles generated are mostly regular, but the shape edges are not reconstructed perfectly, as seen at the top of the sail. This is an issue, because the length of the sail border is an important parameter that is used to validate a good sail reconstruction. In order to achieve a mesh with better edges, the neighborhood is expanded until $\mu = 5$. Even though the maximum edge length is set to $L_{max} = 0.2$ to avoid large triangles, the resulting triangulation is more chaotic, as visible in Figure 3.62.c. Here, the imperfection on the border are avoided, to the detriment of a worse visualization. Their differences can be visualized through a comparison of their edge length and angles, as explained above. From Figure 3.63, as expected, the first triangulation have most angles between 45 and 90 degrees (blue data), while the second one have more scattered values (red data). Also the edge length analysis confirm the better regularity of the first mesh, in fact its triangles have edges length ratio centered around one, which means that most triangles are composed by edges with more or less the same length.

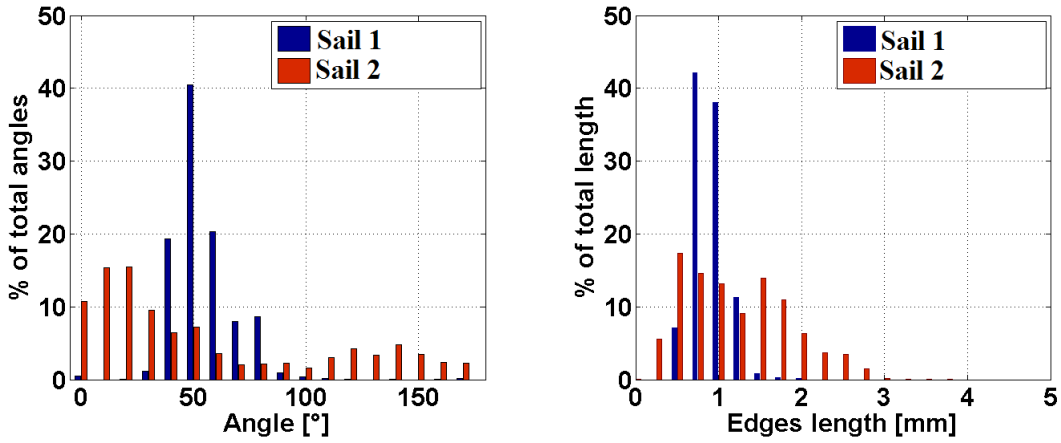
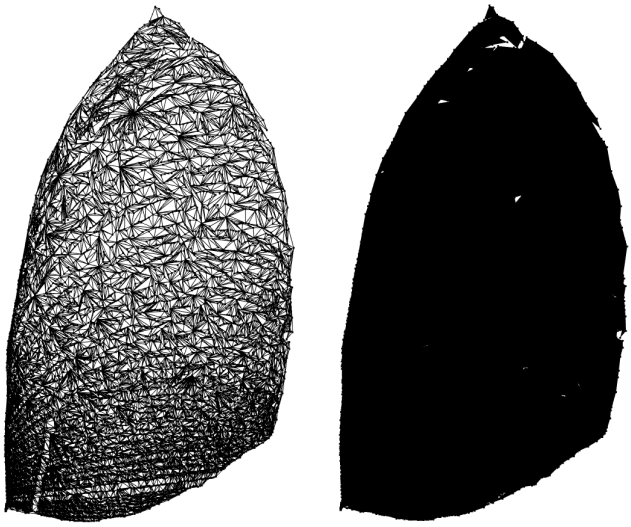


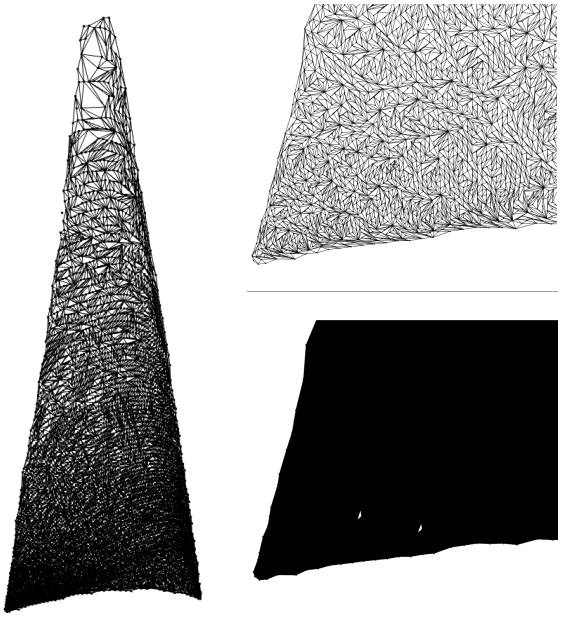
Figure 3.63: Sail model triangulation comparison - blue data refers to Figure 3.62.b, red data to Figure 3.62.c

The determination of the best result is not absolute, but has to be referred to the final goal of the reconstruction. In this case the only analysis of the properties of the mesh is not sufficient. A more regular mesh could be discarded, because it does not represent well the sail on its border. These considerations have to be made also for the triangulation of real acquisitions. Real data sets are different from the

model set for a main issues: the point density is not constant (i.e. points are not equispaced). This is due both to the presence of noise and, most important, to the fact that the density varies with the acquisition angle (see section 2.4.3) and lead to a cloud with high density on the bottom part and sparser points on the top. Loss of data could also happen in the top, as referred in section 2.4.3.



(a) Gennaker triangulation



(b) Mainsail triangulation

Figure 3.64: Greedy Projection: Real acquisition triangulations

Also the final mesh is affected, because it would be composed of small triangles

in the bottom and bigger ones on the top. A good mesh is however not retrieved, since there is a part of the cloud which have less samples than the other. Tests on real acquisitions are performed on a gennaker acquired in wind tunnel and on a mainsail acquired on field. Even with low level constraint, the triangulations created are not acceptable, due to the presence of holes. Missing triangles cannot be seen in the triangulation visualization, showed in Figure 3.64.a and 3.64.b left, but are evident in Figure 3.64.a and 3.64.b right, where the area of each triangle has been filled, to highlight empty spaces. The parameters used for the creation of the meshes showed in Figure 3.64 are displayed in Table 3.32.

	# Neigh.	μ	Max. Edge [a.u]	Min. Angle [°]	Max. Angle [°]	Max. Surface Angle [°]
Gennaker	50	7	300	30	150	20
Mainsail	50	8	400	45	150	20

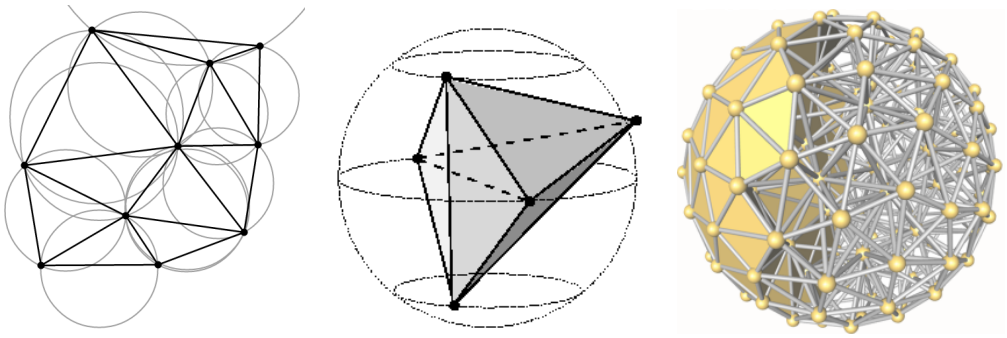
Table 3.32: Greedy Projection: Real acquisition triangulations parameters

Since the absence of holes is the first prerequisite that the triangulation has to satisfy to be considered and it is violated, these meshes are not acceptable. Changing the parameters does not lead to any improvement in the triangulations and other tests on different acquisition showed that it is not possible to fully reconstruct a real sail acquisition.

3.5.2 Delaunay Triangulation

The Delaunay triangulation is a special case of a regular triangulation, such as there is no point from the input set **P** inside the circumcircle of any triangle. The triangulation is named after Boris Delaunay for his work on this topic in 1934 [Delaunay, 1934]. Delaunay triangulation is born as a two dimensional algorithm, but the direct three dimensional approach is already been defined [Mücke, 1993], where triangles are replaced by tetrahedra and circles with spheres. Three dimensional Delaunay triangulation are then defined as a set of tetrahedra where no points from the input set lays inside the circumsphere created by each tetrahedron.

The other property of these triangulations are that they maximize the smallest angle between two incident edges and minimize the largest circumference of each triangle [Sloan, 1993], [Fortune, 1992]. For a set of points on the same line there is no Delaunay triangulation (in fact, the notion of triangulation is undefined for



(a) 2D Delaunay Triangulation

(b) 3D Delaunay Triangulations

Figure 3.65: Delaunay Triangulation Visualization

this case). For four points on the same circle (e.g., the vertices of a rectangle) the Delaunay triangulation is not unique: clearly, the two possible triangulations that split the quadrangle into two triangles satisfy the Delaunay condition. From a mathematical point of view the Delaunay triangulation is the dual graph of the Voronoi tessellation [Mücke, 1993], that is a special kind of decomposition of a metric space, determined by distances to a specified discrete set of points in the space. In the simplest and most common case, a set of points $\mathbf{S}$ in the plane, the Voronoi diagram for $\mathbf{S}$ is the partition of the plane which associates a region $\mathbf{V}(\mathbf{p})$ with each point $\mathbf{p}$ from $\mathbf{S}$ in such a way that all points in $\mathbf{V}(\mathbf{p})$ are closer to $\mathbf{p}$ than to any other point in $\mathbf{S}$. Lots of works in literature cover the Delaunay triangulation topic and a large number of algorithms have been proposed, which can be subdivided in three main categories: Divide and Conquer methods, Incremental methods and Sweep methods.

Divide and Conquer methods

The divide and conquer algorithms require to know in advance the number and the values of the data points. Firstly, the point region is recursively divided in almost equal sub-regions, then, the connectivity information for each sub-region is produced and finally, these results are merged to get the overall triangulation. An example is provided by [Guibas and Stolfi, 1985].

Incremental methods

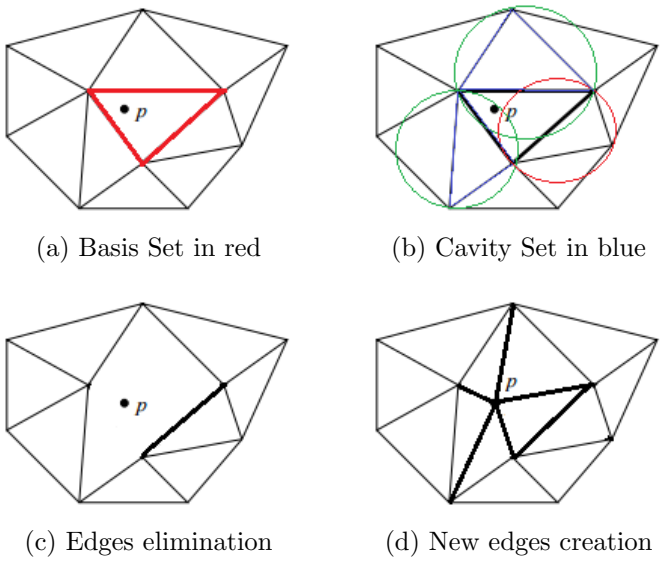
Several incremental algorithms have been implemented: Algorithm 3.7 describes the one simultaneously developed by [Watson, 1981] and [Bowyer, 1981], because it is implemented in the CGAL libraries. At first the algorithm constructs a simplex (triangle in 2D, tetrahedron in 3D) enclosing all the data set points, the so-called

bounding triangulation. Then each point, one by one, is injected into the current triangulation, maintaining the invariant that the triangulation is Delaunay, following the steps described.

Algorithm 3.7 Incremental method Delaunay triangulation algorithm

1. Creation of the bounding triangulation
2. For each point $\mathbf{p_i}$ not in the boundary triangulation:
   (a) Creation of the basis set (all simplicies enclosing $\mathbf{p_i}$) and the cavity set (all simplices whose circumcicle contains $\mathbf{p_i}$) of the point $\mathbf{p_i}$
   (b) Elimination of the edgs that belong to the basis and the cavity
   (c) Connection of $\mathbf{p_i}$ with all the points belonging to the cavity contour

Figure 3.66 illustrates the steps performed to add a point p in an already formed Delaunay triangulation. In 3.66.a the basis set is highlighted, while in 3.66.b the cavity set is shown in blue. It is composed of the simplicies whose circumcircle (in green) contains **p**.



(a) Basis Set in red (b) Cavity Set in blue

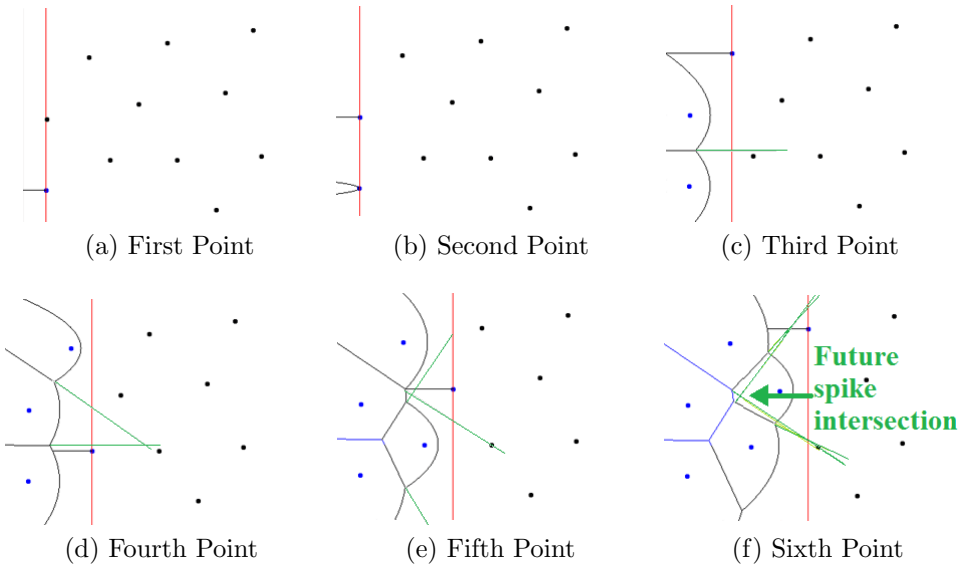
(c) Edges elimination (d) New edges creation

Figure 3.66: Incremental method Delaunay triangulation visualization

Sweep methods

They are based mainly on the work of [Fortune, 1987] and [Seidel, 1988]: they construct the Voronoi tessellation in the plane by moving a line on it, called sweep

line, and the forms the Delaunay triangulation starting from the created diagram. A very good explanation of these algorithms is provided by [Aurenhammer, 1991] and it is reported in the following. Let's consider the construction of the Voronoi diagram of a set of points, which is performed in several steps. For each step the Voronoi diagram is created only for the points sites to the left of the sweep line H and of H itself, considered as an additional site. The first steps for a random set of 2D samples are showed in Figure 3.67. Because the bisector of a line and a non incident point is a parabola, the boundary of the Voronoi region of H is a parabola in the case of a single point (see Figure 3.67.b) or a connected chain of parabola segments whose top and bottommost edges tend to infinity. This chain is called the wavefront W (see Figure 3.67.c and following).



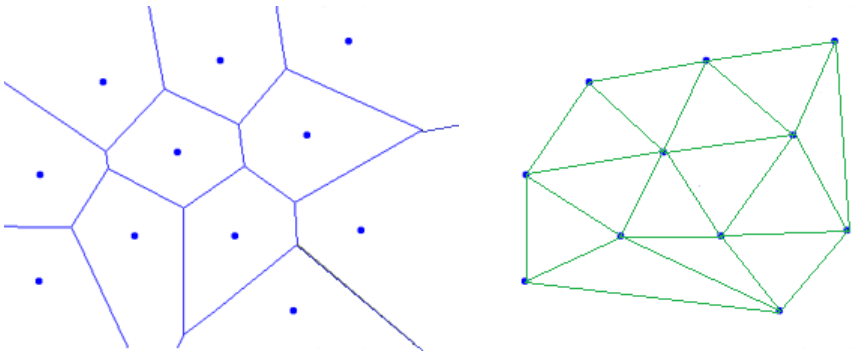
(a) First Point (b) Second Point (c) Third Point

(d) Fourth Point (e) Fifth Point (f) Sixth Point

Figure 3.67: Sweep method visualization: creation of the Voronoi diagram

As the sweep line moves on to the right, the waves must follow because the sets of points added to the diagram grows. On the other hand, each Voronoi edge to the left of W that currently separates the regions of two point sites p_i ,p_j will be (part of) an edge in the final Voronoi diagram V(S). During the sweep, there are two types of events that cause the structure of the wavefront to change, namely when a new wave appears in W, or when an old wave disappears. The first happens each time the sweep line hits a new site. At that very moment of the contact B(H; p), where B is the bisector operator, is a horizontal line through p (see Figure 3.67.a), a little later its left half-line unfolds into a parabola (see Figure 3.67.b).

New parabolas must be inserted into the wavefront, if existing, by gluing it onto the wave of p (see Figure 3.67.c for per-gluing and Figure 3.67.d for post-gluing). Let p, q be two point sites whose waves are neighbors in W. Their bisector, B(p; q), gives rise to a Voronoi edge to the left of W. Its prolongation into the region of H is called a spike. In figure 3.67, spikes are depicted as gray lines; one can think of them as tracks along which the waves are moving. A wave disappears from W when it arrives at the point where its two adjacent spikes intersect. Its former neighbors become now adjacent in the wavefront. As soon as all point sites have been considered and all spike intersections have been processed, V(S) is obtained by removing the wavefront and extending all spikes to infinity. Once the Voronoi diagram has been created, it can easily be converted in the Delaunay triangulation connecting neighboring cells points.



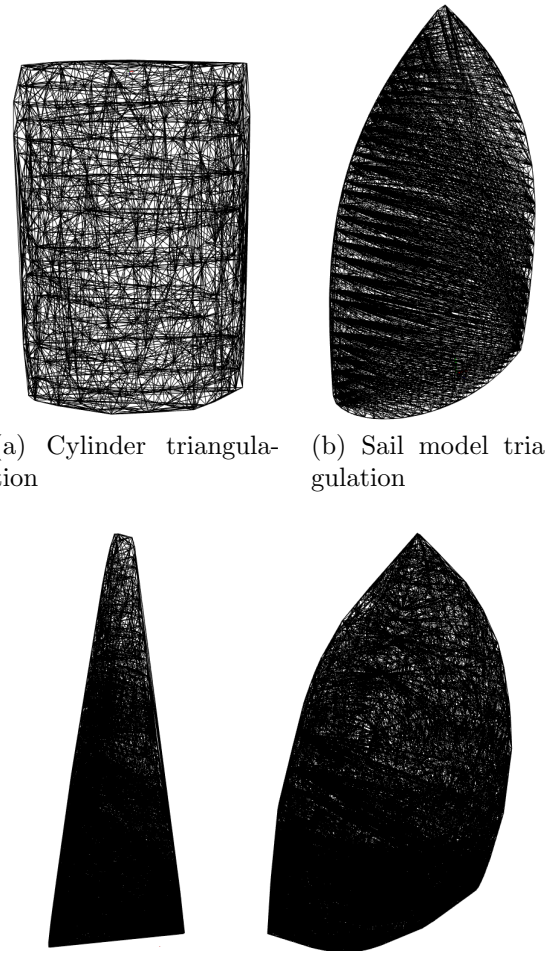
(a) Final Voronoi diagram from Figure 3.67

(b) Correspondent Delaunay triangulation

Figure 3.68: From Voronoi diagram to Delaunay triangulation

As done for the Greedy Projection algorithm the performances of the Delaunay Triangulation are tested through a series of shapes, starting from a planar surface and getting closer to real acquisitions. The data sets used are explained in the previous section. The native three dimensional Delaunay is applied for all the following tests. The plane set fails to be reconstructed, because it violates one of the assumption of the 3D Delaunay, which is exactly that four points do not have to lie on the same plane. Even without this hypothesis, trying to triangulate a two dimensional set with a three dimensional algorithm, is not suitable option. Since the algorithm does not have any parameters, the reconstruction is directly performed on any shape chosen and the results are displayed in Figure 3.69. The extremely unclear visualization of every shape is due to the fact that the 3D algorithm creates

tetrahedra, so there are a many faces to be visualized. The procedure does not recreate the surface of the sail, but instead the volume occupied by the point in the space. The geometric object created correspond to the convex hull of the data set. Mathematically the convex hull of a set of points **S** in the Euclidean plane/space defined as the smallest convex set that contains **S** [De Berg et al., 2000]. A quick solution consists in eliminating from the mesh all the triangles which have at least one edge longer then a specified threshold. With this filtering the mesh remains a triangulation, since remaining triangles are not modified.



(a) Cylinder triangulation

(b) Sail model triangulation

(c) Mainsail triangulation

(d) Gennaker triangulation

Figure 3.69: 3D Delaunay: synthetic shapes triangulations

If the point of the clouds are uniformly distributed on the surface the elimination should leave all the triangles that correspond to the sail external surface, while

deleting faces that lie inside the sail. This strategy, however, is absolutely not robust, because the presence of noise could slightly move the samples inside or outside the ideal surface. This lead to the possibility of the creation of very small tetrahedra along the surface, since there are no lower limit to their interior solid angle. Moreover, since in real acquisitions, the point density is not constant, a value of the maximum edge threshold that filter well the triangles on the bottom part of the sail would eliminate triangles, even the ones belonging to the surface, on its top part. In conclusion, three dimensional Delaunay triangulations are unable to reconstruct correctly the surface of a sail, because they are formed by tetrahedra, which are difficult to prune in order to retrieve a surface.

3.5.3 Poisson Surface Reconstruction

The Poisson Surface Reconstruction algorithm expresses the problem or surface reconstruction, starting from a set of oriented points, as the solution of a Poisson equation. Poisson's equation is a partial differential equation of elliptic type; it has many uses, for example describes the potential energy field caused by a given charge or mass density distribution. Mathematically it is defined as:

$$\triangle\varphi = f \tag{3.41}$$

where $\triangle$ is the Laplace operator and φ and f are real or complex-valued functions on a manifold. f is a known function and φ is the function that need to be retrieved. In the three dimensional space, the equation takes the form:

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right)\varphi\left(x, y, z\right) = f\left(x, y, z\right) \tag{3.42}$$

The Poisson equation, with $f = 0$ becomes the Laplace equation. The use of the Poisson equation to reconstruct three dimensional surfaces has been formulated and implemented first by Kazhdan et al. in [Kazhdan et al., 2006]. They compute a 3D indicator function χ, which is defined as 1 for points inside the model and 0 outside and then obtain the reconstructed surface by extracting an appropriate isosurface. The key step here is the presence of an integral relationship between oriented points, sampled from the surface to be retrieved, and the indicator function of the model. Specifically, the gradient of the indicator function is a vector field that is zero almost everywhere, except near the surface, where is equal to the surface

normals.

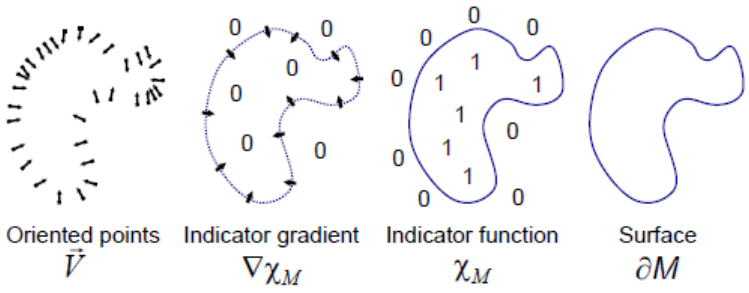


Figure 3.70: Intuitive illustration of Poisson reconstruction in 2D

In their implementation Kazhdan et al. convolute the indicator function with a smoothing filter and consider the gradient field of the smoothed function, in order to avoid unbounded values at the surface boundary. Again, the gradient of the smoothed function is equal to the vector field obtained by smoothing the surface of the normal field. The demonstration is here omitted, but shown in [Kazhdan et al., 2006]. The equality can be formalized as:

$$\nabla\left(\chi_M * \tilde{F}\right)(q_0) = \int_{\partial M} \tilde{F}_p(q_0)\overrightarrow{N_{\partial M}}(p)dp \tag{3.43}$$

where ∂M is the boundary of the solid M (source of the samples), χ_M is the indicator function of M, $\overrightarrow{N_{\partial M}}(p)$ is the inward surface normal at a point $p \in \partial M$, $\tilde{F}(q)$ is a smoothing filter and $\tilde{F}_p(q) = \tilde{F}(q-p)$ its translation to the point p. The problem of computing the indicator function is thus reduced to an inversion of the gradient operator, i.e. finding the scalar function χ whose gradient best approximates a vector field $\overrightarrow{V}$ defined by the samples. The surface ∂M can't be retrieved from the integral (3.43), because the surface geometry is yet unknown. The integral is then approximated with a discrete summation, using the point set to partition ∂M into patches.

$$\nabla\left(\chi_M * \tilde{F}\right)(q) =$$

$$= \sum_{s\in S}\int_{P_S} \tilde{F}_P(q)\overrightarrow{N_{\partial M}}(p)dp \;\cong\; \sum_{s\in S}|P_S|\,\widetilde{F_{s\cdot p}}(q)s.\overrightarrow{N} \;\equiv\; \overrightarrow{V}(q) \tag{3.44}$$

In (3.44) s.p. denote the sample point, s.N its sample normal and P_S the patch

around it. After the determination of a vector field $\overrightarrow{V}$, the following step is finding the solution of the function $\nabla\widetilde{\chi}$ such that $\nabla\tilde{\chi} = \overrightarrow{V}$. $\overrightarrow{V}$ is generally not integrable, which means that an exact solution does not exist, so the solution is a least-squares minimization. To find it, the divergence operator is applied both terms and thus the Poisson equation $\triangle\widetilde{\chi} = \nabla \cdot \overrightarrow{V}$ is formed. The vector field can be defined differently in two situations, depending if the samples are uniformly distributed or not. For its explicit definition the reference is [Kazhdan et al., 2006], it is composed starting from an interpolation of the smoothed normals in each patch. The solution of a Poisson equation is a well studied problem, with optimized matricial implementation, which is not addressed in this work. Solving the equation returns the indicator function $\widetilde{\chi}$, which is evaluated in many points in the Euclidean space (see Figure 3.70 for a 2D illustration). Its values vary from zero (points far from the surface) to one (points exactly on the surface), intermediate values reflect the distance of the point from the surface. The final step is to extract an isosurface, correspondent to a value, γ. The output surface $\partial\tilde{M}$ can be then defined as:

$$\partial\tilde{M} = \left\{q \in \mathbb{R}^3 | \tilde{\chi}(q) = \gamma\right\} \tag{3.45}$$

It is evident that $\partial\tilde{M}$ is an approximation of ∂M in every case when $\gamma \neq 1$. The implementation of the Poisson Surface reconstruction algorithm is made through the PCL libraries, with the Poisson template. The input given is a point cloud with its associated normals, that have to be computed before starting the procedure. Some parameters can be set to specify the desired accuracy, otherwise the algorithm works with standard values. During the test, the influence of the main parameters has been investigated and are explained below, although the results displayed are the ones with the best output found. The cloud used are the same of the previous sections tests. An initial test is the creation of a plane from a grid of points, Figure 3.71 shows the resulting surface.

The very first difference from the previous algorithms is that, in this case, the mesh does not pass through the input points. This happens because the mesh is extracted as an isosurface from the indicator function; the points (or vertices) of the mesh thus correspond to the spacial coordinates where this function has been evaluated. Another visualization difference is that the mesh is no more triangular, but composed of quadrilateral shape. This is due to the fact that the PCL implementation divides the Euclidean space with an octree structure (see section 3.1.8)

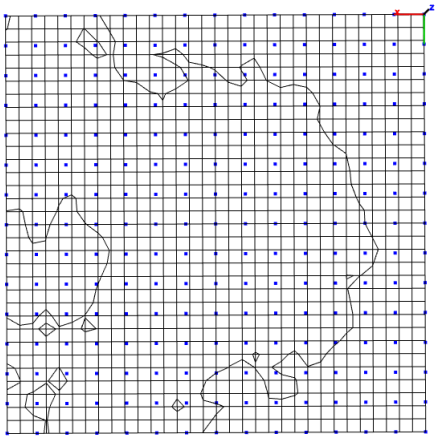


Figure 3.71: Poisson Surface reconstruction: plane - blue points are the input data set

at the beginning of the algorithm, to reduce the computational calculus complexity. Since the octree is a cube-based space subdivision, the evaluation of the indicator funcion takes place on the corner of the cubes, creating a surface formed by squares. The accuracy of the mesh can be manipulated through some parameters:

- Depth of the octree : the maximum depth of the octree. Higher values of this parameter result in the division of the space in smaller cubes, leading to a higher resolution of the output mesh.
- Scale : the ratio between the diameter of the cube used for reconstruction and the diameter of the sample bounding box. The user can choose if the reconstruction of a patch can take place with a cube that has its same size or a bigger one. In other words the surface generated can be enlarged by a multiplying factor.
- Solver Divide : depth of the octree at which the Laplacian equation is solved (recommended equal to the depth of the octree).
- Iso Divide : depth of the octree at which the isosurface is extracted (recommended equal to the depth of the octree).

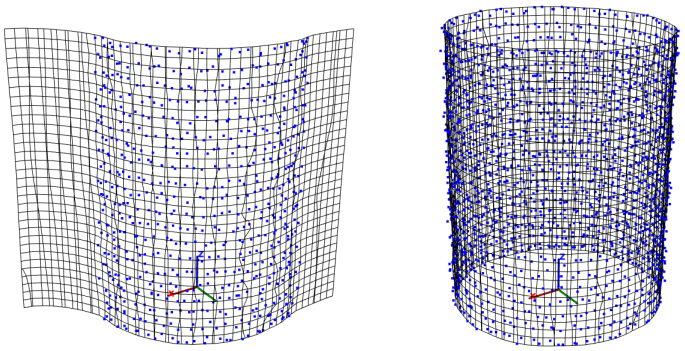
For the mesh displayed in Figure 3.71 the parameters are set as visible in Table 3.33.

These parameters are used as standard and are the ones recommended in the PCL libraries and in [Kazhdan et al., 2006]. The next test is performed with a

Depth	Scale	Solver Divide	Iso Divide
8	1	8	8

Table 3.33: Poisson Surface reconstruction: plane reconstruction parameters

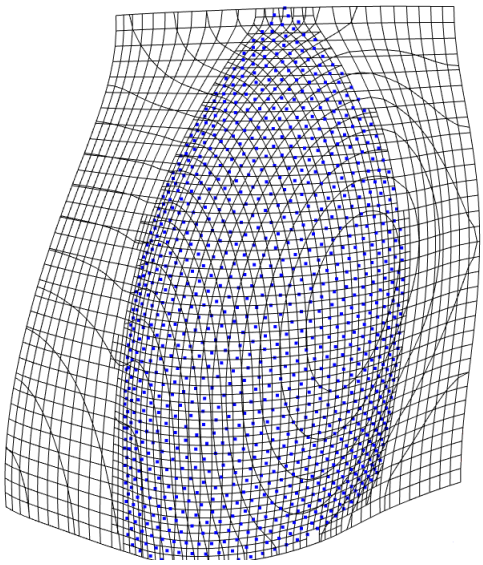
noisy cylinder, cut to show only 60 degrees of its surface, the reconstructed surface is showed in Figure 3.72.a.



(a) Cut Shape (b) Full Shape

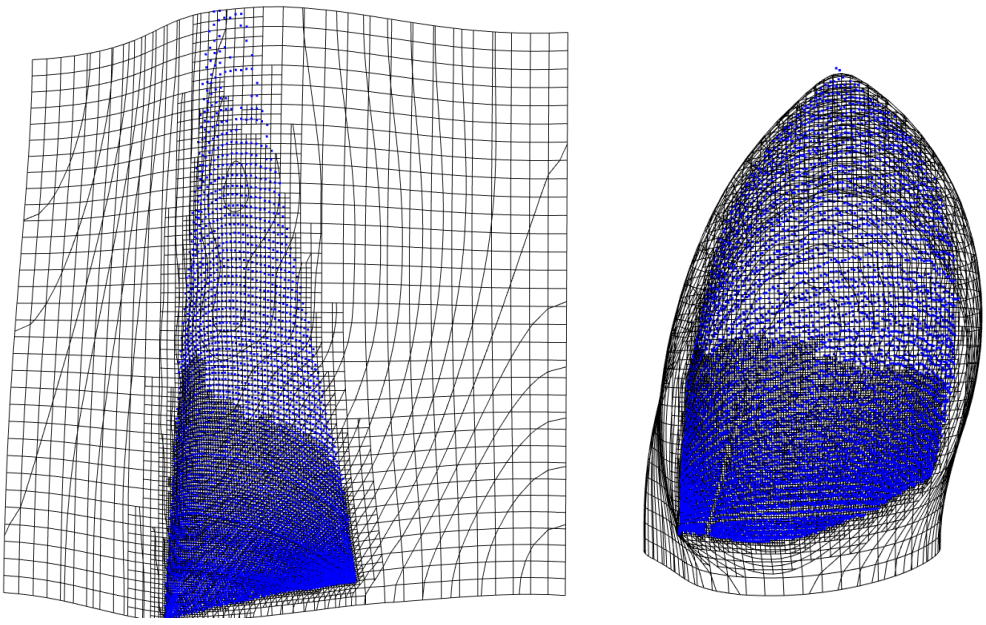
Figure 3.72: Poisson Surface reconstruction: noisy cylinder - blue points are the input data set

The resulting mesh is not fully adherent to the input point cloud, but it has "wings" on its sides. This behavior happens because the indicator function is defined in the whole Euclidean space, even outside of the shape and in the final step of the algorithm, the isosurface is extracted from it. The wings creation can be explained looking at the 2D visualization of the indicator function (Figure 3.70); its value can space from 0 to 1 depending if a point is inside (1), outside (0) or near the line/surface. If the data set does not represent a closed set, the isosurface indicator function does not have a clear distinction between the inside and outside, so the surface extraction can return a wrong shape. To prove the good behavior of the Poisson reconstruction with closed data sets, the full cylinder surface is showed in Figure 3.72.b. However, since sails are not a closed data set, the Poisson surface reconstruction algorithm returns similar results with the model sails and the real acquisitions, which are showed in Figure 3.73 and Figure 3.74.



(a) Model sail

Figure 3.73: Poisson surface reconstruction: model sail - blue points are the input data set

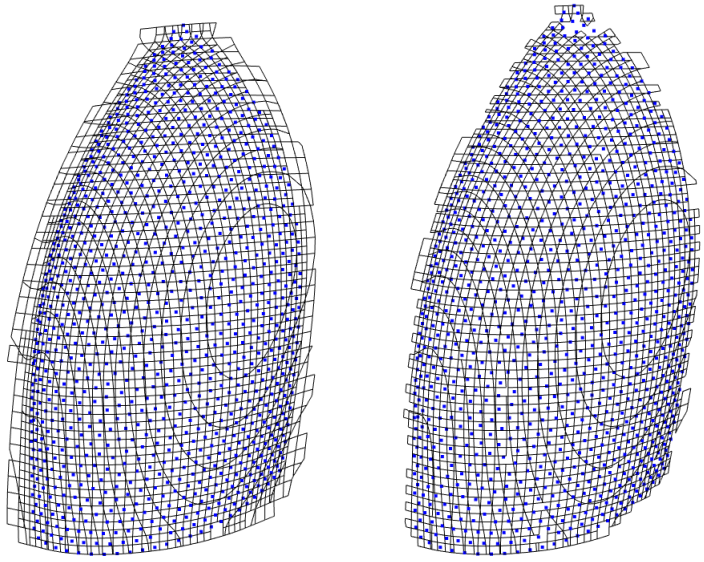


(a) Mainsail (b) Gennaker

Figure 3.74: Poisson surface reconstruction: real sails - blue points are the input data set

The surfaces reconstructed have some properties that would fit perfectly for the sail surface: the smoothness and the adaptability to clouds with variable point density. In Figure 3.74 faces of different dimensions are visible; on the bottom part

of the sail, where the density is higher, faces are smaller, but on the top, where points are sparser, the squares are bigger. Note that, due to the particular structure of the octree, four little squares fit exactly in a big square. Smoothness come from having approximated the input data set, instead of interpolating it. As done for the Delaunay triangulation, a quick attempt at retrieving a correct final mesh has been tried. In this case, the mesh topology is fine, as there is no need to check its edges or angles. In order to try to get rid of the unwanted extensions of the mesh, faces whose vertices are not close to the input data set are deleted. Remember that the vertices of the mesh do not correspond to the input points, so, to determine if a face has to be kept, a distance threshold has to be defined. Specifically, a face is not filtered if all of its vertices have an input point closer than the distance threshold. The results of the filter on the model sail mesh for two different distance thresholds are showed in Figure 3.75.



(a) $d_{max} = 0.1$ (a.u.) (b) $d_{max} = 0.05$ (a.u.)

Figure 3.75: Sail model mesh filtered

In both cases the filter can't retrieve a correct shape. In the first try (Figure 3.75.a) the threshold is too high and many faces not belonging to the real surface still remain. On the other try the distance limit is so low that some faces that should remain are deleted, while at the same time, some external shapes still remain. It is evident that this simple filter is not able to reconstruct correctly the sail surface and if it does not work on the model data set it is hardly probable that it would

work on real acquisitions, which are not uniformly sampled. The filter has been tested also on the mainsail and gennaker meshes, where it did not work correctly, as expected.

3.5.4 Trimmed B-spline surfaces

Another common way to represent geometry in computer graphics, CAD and computer vision are B-spline (Basis-spline) curves and surfaces as well as their abstraction to Non-Uniform Rational B-Splines (NURBS). They are used in a wide field of applications including entertainment industry, mechanical, electrical and medical engineering, architecture and computer vision. In the NURBS fundamental book [Piegl and Tiller, 1996], the concept of B-spline curves and surfaces as well as NURBS are described. B-spline curves are introduced, then their 3D generalization, B-spline surfaces, are explained. The first element to be introduced is the knot vector $\Xi = \{\xi_1, \xi_2, .., \xi_{n+p+1}\}$, which is a non-decreasing set of coordinates in the parameter space $\Omega_c = [\xi_1, \xi_{n+p+1}] \subset \mathbb{R}$ where n is the number of basis functions used for the B-spline curve and p is the polynomial order. The knots partition the parameter space into elements. Through the knot vector in hand, the B-spline functions are defined with the Cox-de Boor recursive formula [Cox, 1972], [De Boor, 1972]:

$$N_{i,0}(\xi) = \begin{cases} 1 & if\ \xi_i \leq \xi \leq \xi_{i+1} \\ 0 & otherwisw \end{cases} \tag{3.46}$$

for p=0, while for p=1,2,3,.. they are defined by:

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p-1} - \xi_{i+1}} N_{i+1,p-1}(\xi) \tag{3.47}$$

The basis constitutes a partition of a unity, that is, $\forall \xi$,

$$\sum_{i=1}^{n} N_{i,p}(\xi) = 1 \tag{3.48}$$

Also, the basis function is point-wise, non-negative over the entire domain. Another important feature is that each p-th order function has p-1 continuous derivatives. A B-spline curve $c(\xi) : \Omega_c \to \mathbb{R}^3$ with parametric domain $\Omega_c \subset \mathbb{R}$ is constructed by linear combination of B-spline basis functions. Given n basis functions

$N_{i,p}$with i=1,2,..,n, the n vector-valued coefficients of the basis function (called control vector) $b_i \in \mathbb{R}^3$ defines the curve as:

$$c(\xi) = \sum_{i=1}^{n} N_{i,p}(\xi) b_i \tag{3.49}$$

The idea is to manipulate the B-spline curve $c(\xi)$ by changing the values of the control points b_i. Fitting a B-spline curve to a set of data points p is the task of finding values for the control points that minimize the distance between p and c(ξ) as shown in Figure 3.76.

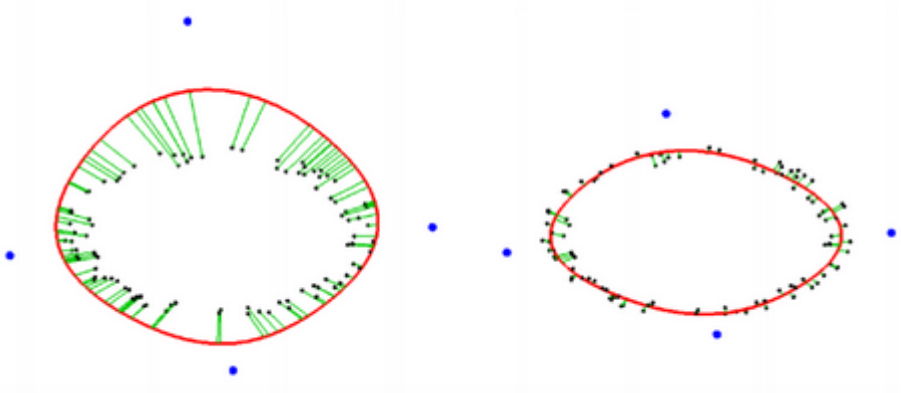


Figure 3.76: Fitting B-spline curves: the distance (green) between the points and the closed B-spline curve (red) is minimized by manipulating the position of the control points

The definition of a B-spline surface follows the one of the B-spline curve. A B-spline surface $S(\xi) : \Omega_S \rightarrow \mathbb{R}^3$ with a parametric domain $(u, v) \in \Omega_S \subset \mathbb{R}^2$ is constructed by linear combinations of the tensor product of B-splines basis functions. Given n, m basis function $N_{i,p}$ and $M_{j,q}$ with i=1,2..,n and j=1,2,..m, the vector-valued coefficients, called control grid, $B_{i,j} \in \mathbb{R}^3$, defines the surface as

$$S(u, v) = \sum_{i=1}^{n} \sum_{j=1}^{m} N_{i,p}(u) M_{j,p}(v) B_{i,j} \tag{3.50}$$

The same characteristics as for B-spline curves apply for B-spline surfaces. The derivatives in given parametric direction may be determined from the respective one-dimensional basis function. The same concept explained in Figure 3.76 in 2D applies also for the three dimensional case, as visible in Figure 3.77.

Due to the tensor product leading to an orthogonal parametric domain, the

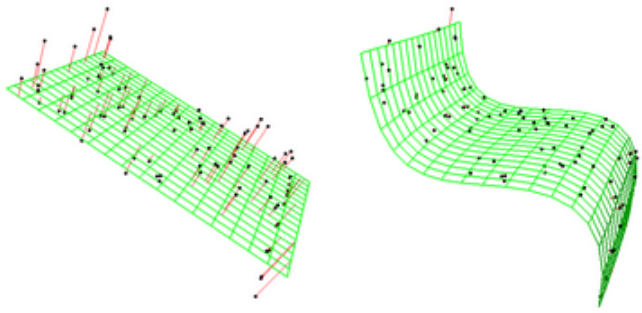


Figure 3.77: Fitting B-spline surfaces: the distance (red) between the points and the B-spline surface (green) is minimized by manipulating the control points

boundaries are four-sided with the properties given by respective basis function of the surface. This is not desirable if the searched shape does not have four sides, which is the case for this work. One possibility to get rid of the four-sided shape of a B-spline surface is to trim away areas that lie outside a certain region. Such a trimming region $\Omega_t \subset \Omega_s$ can be defined on the parametric domain using B-spline curves, the concept idea is displayed in Figure 3.78.

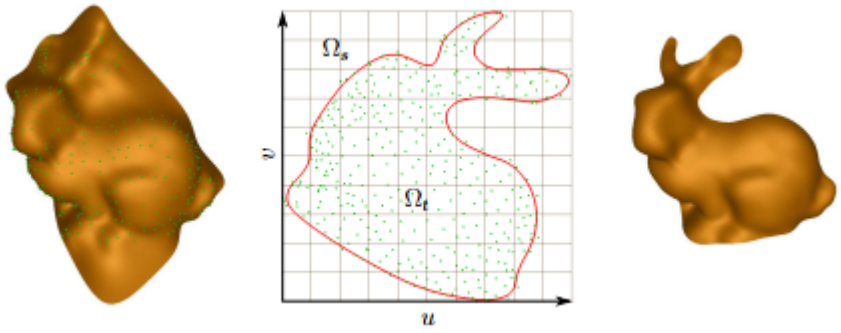


Figure 3.78: B-Spline surface trimming

The code for fitting B-spline curves and surfaces are implemented in the PCL libraries and the behavior is tested with the data sets presented in the previous sections. Fitting a curve on a set of points is done iteratively: first a minimum number of control points is considered and the correspondent B-spline is created. The curve is initialized on the plane formed by the two biggest eigenvectors, calculated from the principal-component analysis (PCA) of the data. A distance metric is associated to the curve, the simplest one is the point-to-line distance, and a threshold is defined. If the evaluation of the metric for a B-spline does not satisfy the limit, then the control points are moved in order to minimize the cost function (see Figure

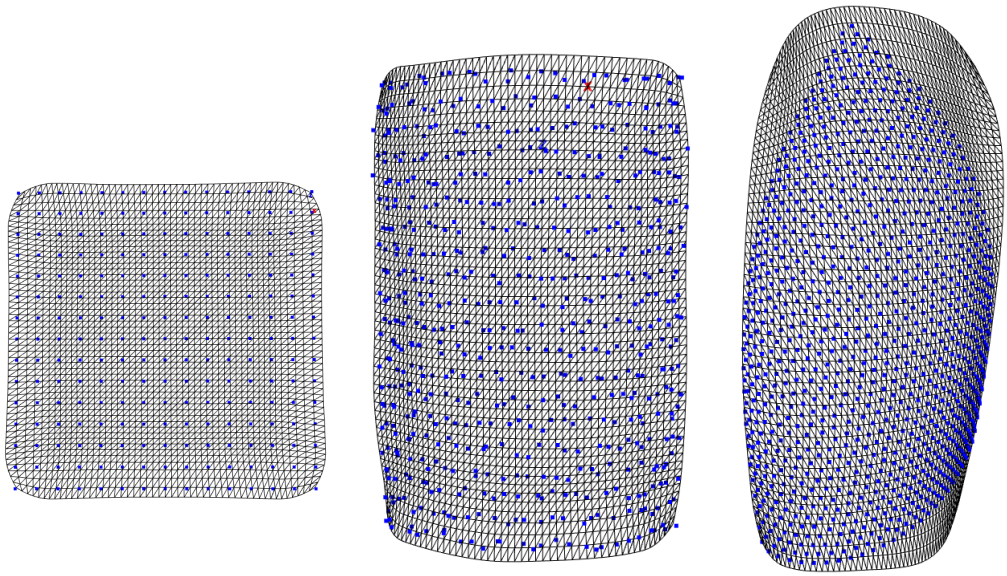
3.76 and Figure 3.77). Additionally, every k iterations, a limited number of control points can be added. Adding control points helps in reconstructing complex or sharp shapes, but increases the computational costs of the procedure. The same steps are followed to reconstruct a B-spline surface. The creation of the surface asks for 6 parameters:

- order: the polynomial order of the B-spline surface, for surfaces as the ones we are interested in, the order is low, less then 3).
- iterations: number of iterations that are performed.
- refinement : maximum number of iterations when additional control points are inserted. For each refinement iteration the control points number is doubled in each parametric direction.
- mesh resolution: number of vertices in each parametric direction. After that the surface is created, a triangulation is created for visualization. This value influences only the visualization of the resulting surface.
- interior smoothness: smoothness of the surface interior.
- interior weight: weight for optimization of the surface interior.

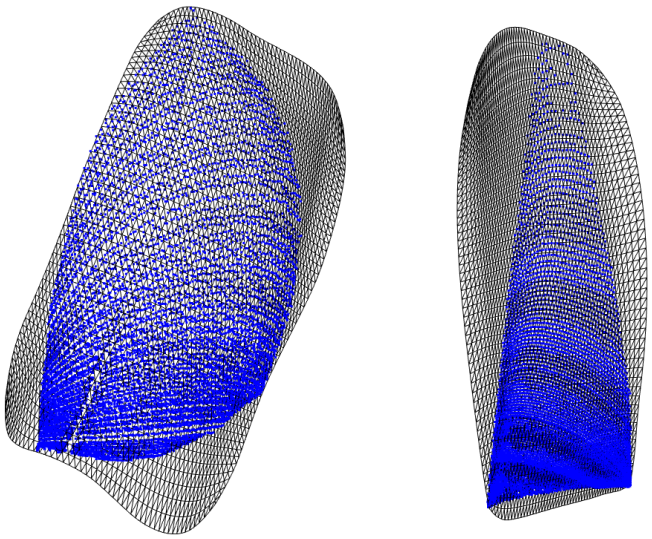
For most application, 10 is a sufficient number of iterations. The surfaces generated from the B-spline are generally an optimal approximation of the data set, however they suffer from the same problem of the Poisson Surface reconstruction. Due to the generation of an approximated surface they have excessive, external parts, not belonging to the real shape, as seen in Figure 3.79. Also, it is possible to notice the four side (although stretched) of the B-spline surface, especially in Figure 3.79.d, that are due to the tensor product used. The parameters used for reconstructing the surfaces shown in Figure 3.79 are shown in Table 3.34.

Order	Iterations	Refinement	Mesh Resolution	Interior Smoothness	Interior Weight
3	10	3	50	0.2	1

Table 3.34: B-spline surface reconstruction parameters



(a) Plane (b) Cut cylinder (c) Model sail



(d) Gennaker (e) Mainsail

Figure 3.79: B-spline surface reconstruction

In this case, differently from Poisson, the trimming of undesired part of the mesh is already implemented and optimized by the PCL libraries, however its use is not as intuitive as the surface generation. In fact, there are ten parameters that needs to be specified:

- control points accuracy: the distance of the supporting region of the curve to the closest data points has to be lower than this value, otherwise a control point is inserted.

- control points iteration: specify the number of iterations without inserting control points.
- maximum number of control points: set an upper bound for the number of control points.
- accuracy: average fitting accuracy of the curve, with respect to the supporting regions.
- iterations: maximum number of iterations performed
- closest point weight: weight for the fitting curve to its closest point (recommended equal to one).
- closest point sigma: threshold for closest points. In other words the algorithm disregards points that are further away from the curve.
- interior sigma: threshold for interior points. In other words the algorithm disregards points that lie within the curve.
- smoothness concavity: value that controls the bending of the curve (0=no bending, <0 inward bending, >0 outward bending).
- smoothness weight: weight of the smoothness term for the curve.

Some parameters are not critical towards the reconstruction of a good trimming curve. They are the maximum iterations number, set to 200, the control points iteration value, which is set equal to three, meaning the the procedure double the number of control points every three iterations and the iterations number, set to 100.

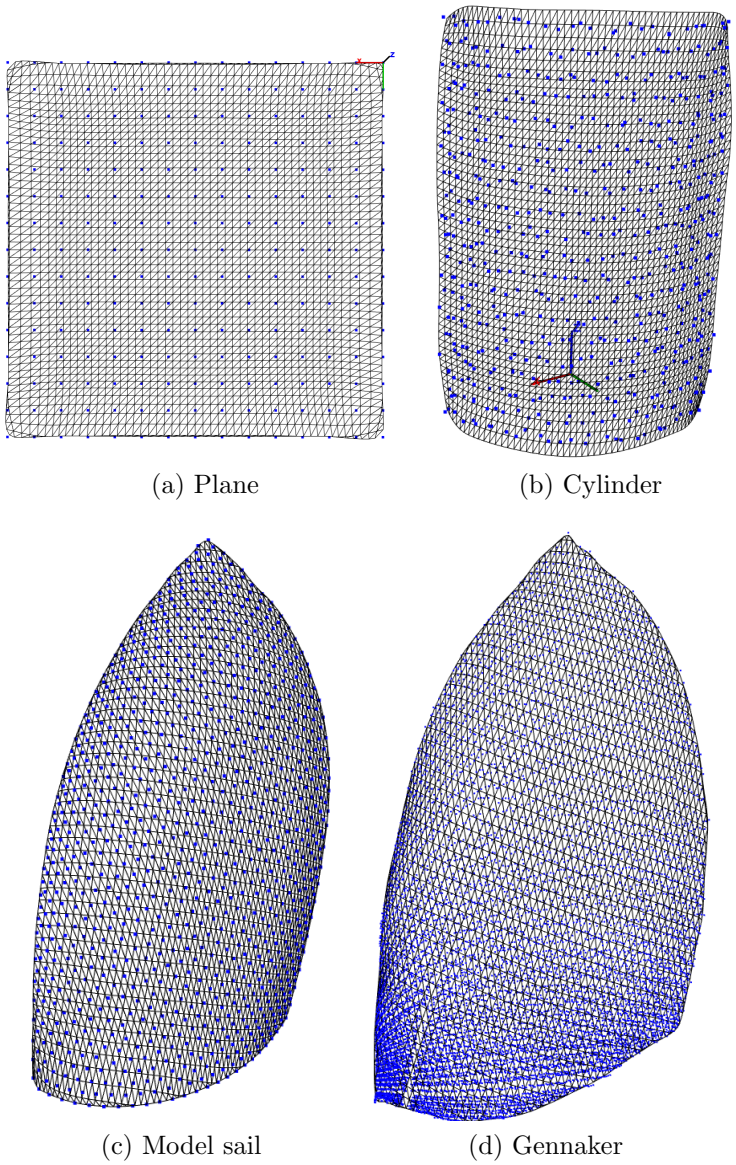
CP accuracy	CP iterations	max CP	Accuracy	Iterations
0.001	3	200	0.001	100

Closest point weight	Closest point sigma	Interior sigma	Smoothness concavity	Smoothness weight
1	0.001	0.00001	1	1

Table 3.35: B-spline surface trimming parameters

The closest point weight, the smoothness concavity and the smoothness weight are all set to their recommended values, which is always one. Tests in changing

their values do not lead to changing in the reconstructed surface. The remaining parameters have to be adapted to the needed accuracy of the trimming and the density of the point cloud. As said many times, in this work a good reconstruction of the border of the sail is critical, so the parameters are set in order to achieve the maximum accuracy possible.



(a) Plane (b) Cylinder

(c) Model sail (d) Gennaker

Figure 3.80: B-spline surface trimming

Searching for higher accuracy lead to use very low parameter values, because the trimming curve has to be really close to the input points, this is done by setting small parameter values, which are displayed in Table 3.35. Even smaller values has

been tested, but the quality of the resulting surfaces would not improve. Actually, after decreasing them of some magnitude orders, the B-spline curve algorithm failed to retrieve any curve at all. This happens because imposing the curve to stay too close to the points neglects the possibility to link close points. Trimming with the presented parameters has been applied on the surface showed in Figure 3.35 and results are showed in Figure 3.80. The mainsail reconstruction is not showed because a proper surface trimming has not been retrieved, with any parameters combination. This is due to the fact the the mainsail acquisition presents holes in the top part, which changes drastically the density of the point cloud in that section. The lack of points lead to the problem explained before, using really low parameters, but happens also for reasonable values (for the other section of the sail considered) and the failure in retrieving an acceptable trimming curve. On the opposite case, relaxing the parameters to avoid this situation, lead to trimming too little of the original surface and thus, the impossibility to use the reconstructed surface for a good representation. Despite the great results showed by the B-spline surface reconstruction, the necessity to have a robust algorithm, which works with many shapes and facing the holes problem, brought to the development of a custom algorithm for the creation of the sail surface.

3.5.5 Reconstruction of sail surfaces

The Delaunay triangulations, is selected to reconstruct the sail surfaces because it proved to be the most robust. However, since the 3D implementations failed, for the reasons explained in section 3.5.2, the key idea is to leave the starting three-dimensional Euclidean space and perform the triangulation on a plane, through the 2D concept of the Delaunay algorithm. In this way, the resulting mesh corresponds for sure to a surface, since it is generated on a two-dimensional domain. It also does not have any problems regarding the creation of tetrahedra that should be trimmed later. Once the 2D mesh is generated, the goal is to reinstate its vertices, which in the Delaunay method correspond to the input points, on the original points, creating a mesh in the three dimensional space. The main steps are described in Algorithm 3.8. The choice of the projection plane is critical, in fact the points distribution in each neighbor can't be altered. This means that, for each sample p, its neighbors in 3D have to correspond to the same neighbors on the plane. For the shapes considered, the best fitting plane and the (α, γ) plane are two planes

that satisfy the criterion. They are, respectively, the plane that minimizes the sum of the squared distances from all the samples to it and the acquisition spherical coordinates plane (for details on the acquisition procedure see section 2.3.1). The cloud projected on the best fit plane is still a three-dimensional data set. In order to transform it in a $n \times 2$ array, which is the input of 2D Delaunay triangulation, it is rigidly rotated on the (x,y) plane . All the points have now a null z coordinate, which can be disregarded.

Algorithm 3.8 Reconstruction of sail surface

1. Project the input points on a plane.
2. Create a new two-dimensional coordinate system (u,v) for the plane and express the data points in it.
3. Performs a 2D Delaunay triangulation.
4. Points in (u,v) are uniquely associated to points in (x,y,z).
5. Triangles in (x,y,z) are created connecting 3D points whose respective are connected in (u,v).
6. Trimming of excessive long edges.

The 2D triangulation, as for the three dimensional case of section 3.5.2, is implemented through the CGAL libraries, with the incremental method. At the end of the algorithm, the resulting object is a triangular mesh, which is an object composed of an array of points and an array of vertices.

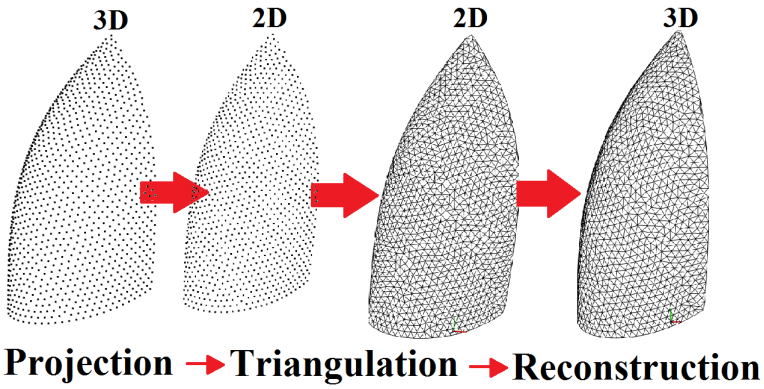
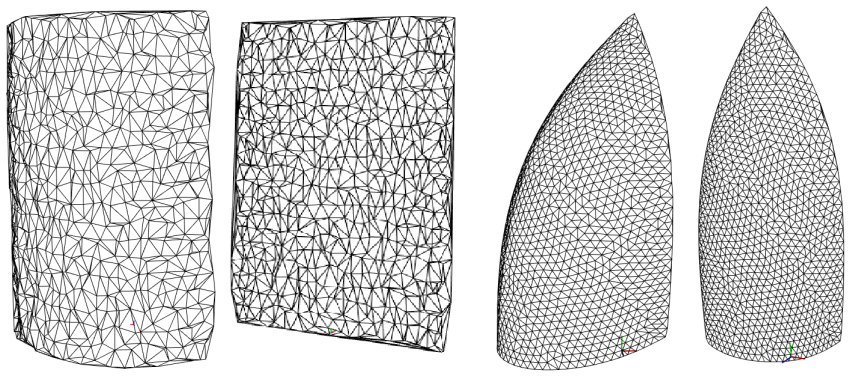


Figure 3.81: Reconstruction of sail surface algorithm overview

The vertices array is a $m \times 3$ array, since the mesh is composed of triangles.

Each entry does not store three points, but three integers, which are the indexes of the samples in the points array. This storage provides a memory optimization and helps greatly in the re-building of the 3D mesh. In fact, given the hypothesis that the i-th point in the 2D cloud correspond to the i-th point in the initial cloud, the final mesh can be created instantiating an empty polygonal mesh object and filling its field with the starting point cloud as points array and the array of indexes retrieved by the 2D Delaunay triangulation as the vertices array. The hypothesis is satisfied by construction, because the projected point clouds are created iterating on the initial set of points and adding them one by one to the projected set after their projection on the considered plane. This procedure works both with the best fit plane or the spherical coordinates plane as projection plane. As for the three-dimensional case, there is no parameter needed for this triangulation, the creation of the mesh is automatic, because user intervention is unnecessary. Unwanted edges along the shape border are present in some meshes, they are linked to the shape overall curvature. The twist of the sail shape could lead, for example, to the connection of the vertices between them. In Figure 3.82.a the vertices of the cut cylinder were linked horizontally.

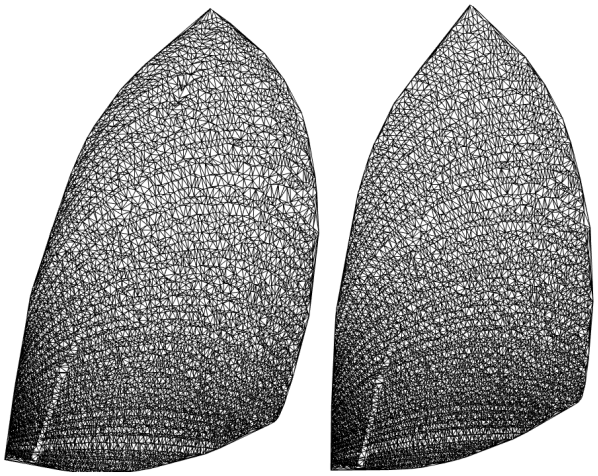


(a) Cut cylinder 3D & 2D meshes (b) Model Sail 3D & 2D meshes

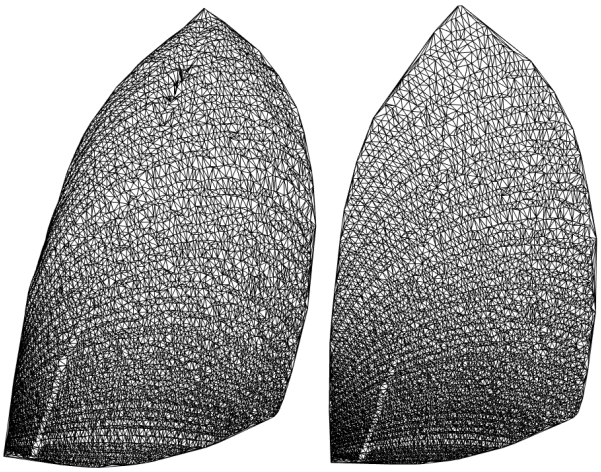
Figure 3.82: Synthetic shape surface reconstruction - Best fit plane

However, these edges are much easier to trim respect to the 3D case, because a single triangle can be checked instead of a whole tetrahedron, which is a combination of triangles, that have to be all evaluated. The plane test, showed in the previous sections, is not representative for this procedure. The cylinder and model sail reconstruction are presented only with the best fitting plane, because their data sets are not coming from the acquisition device and their projection on the spherical

coordinates plane does not make sense.



(a) 3D and 2D meshes with best fit plane

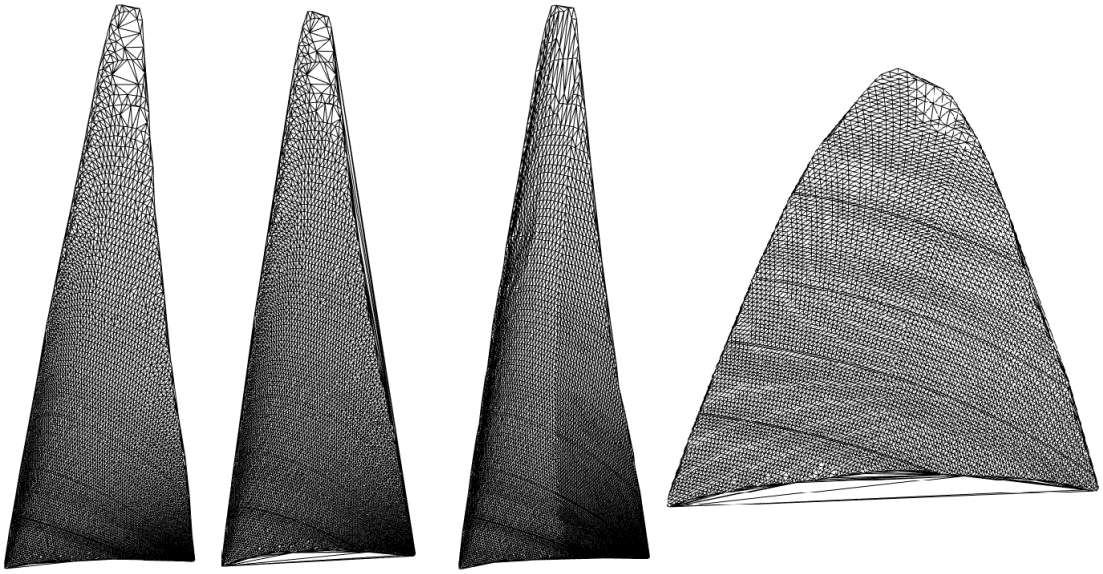


(b) 3D and 2D meshes with (α, γ) plane

Figure 3.83: Gennaker surface reconstruction

The surface retrieved for these two object are showed in Figure 3.82, along with the mesh retrieved on the 2D plane. The more chaotic reconstruction of the cylinder surface has to be attributed to the presence of noise, but through this procedure it is ensured that the union of the triangles form a surface and no tetrahedra are present. For real acquisition the user can choose if triangulating on the best fit plane or on the spherical coordinates plane, the difference in the output meshes are minimal, as seen in Figure 3.83 and Figure 3.84.

The long edges, visible in Figure 3.84.a and 3.84.b in the 2D meshes are successfully eliminated in the trimming step, in fact they are not visible in the 3D meshes. The quality of the meshes are compared using the edge length ratio and the triangle



(a) 3D and 2D meshes with best fit plane (b) 3D and 2D meshes with (α, γ) plane

Figure 3.84: Mainsail surface reconstruction

angles, as explained in section 3.5.2. The comparison is showed in Figure 3.85 for the gennaker, where blue correspond to the (α, γ) plane 3D mesh and red to the best fit plane 3D mesh. The same for the mainsail is showed in Figure 3.86.

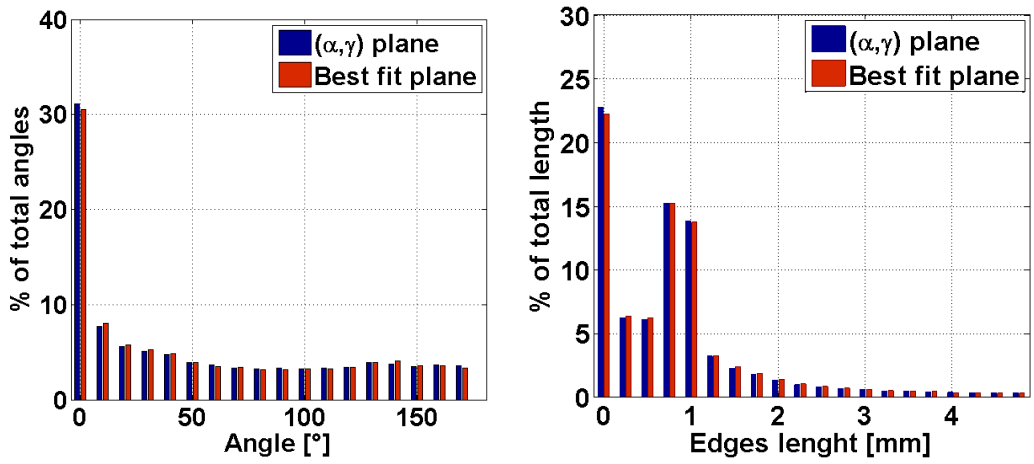


Figure 3.85: Gennaker mesh comparison - blue stands for data of the mesh created in the (α, γ) plane, red in the best fit plane

The comparison shows very little differences, these only parameters can not justify the preference of a method over the other. The choice between using the best fit plane or the spherical coordinates plane is then arbitrary and at the user's

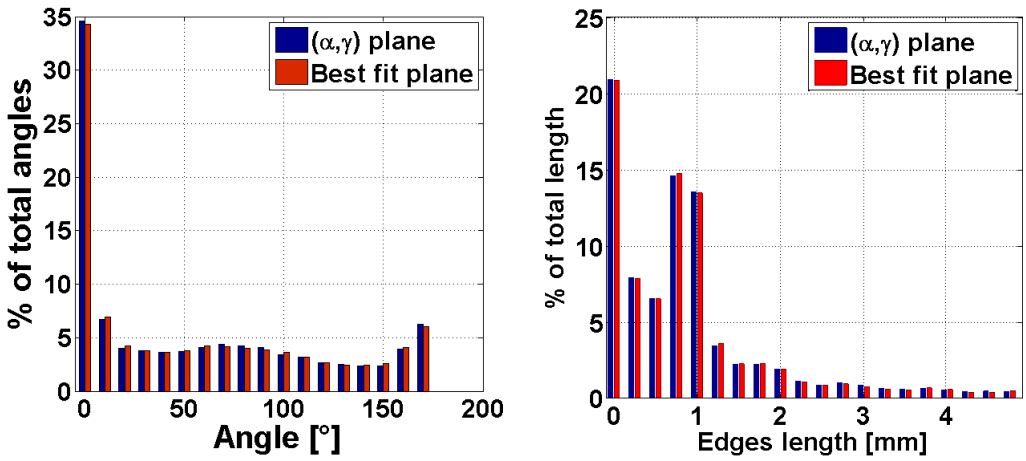


Figure 3.86: Mainsail mesh comparison - blue stands for data of the mesh created in the (α, γ) plane, red in the best fit plane

discretion. Projection on the best fit plane could lead to overlapping projected points, when the sail acquired present heavy twisting during the acquisition.

Chapter 4

Case study

In the present chapter, we report the results of the post process algorithm for two distinct cases: data acquired during a wind tunnel campaign and data acquired during some preliminary tests on field.

All the algorithm steps, starting from the data acquisition and ending with the creation of the mesh, use the techniques selected in the previous chapters. Each operation is discussed highlighting strengths and criticalities encountered.. The softwares developed is presented in detailed and its use explained.

4.1 Software Descriptions

In this paragraph, we present the software dedicated to the acquisition of the data and the one related to the elaboration of the point cloud.

As mentioned in the Introduction, the acquisition of the scans is controlled by a LABView program. Figure 4.1 shows the graphical interface that appears to the user.

This software interface can be considered composed by few principal parts:

1. Motor configuration: enables to establish the communication with the brushless motor;

2. Connection to the scanner: allows for the communication between a personal computer and the lase scanner having the IP address set, using Ethernet connection and TCP-IP protocol;

3. Scanning parameters: where velocity and acceleration of the movement law

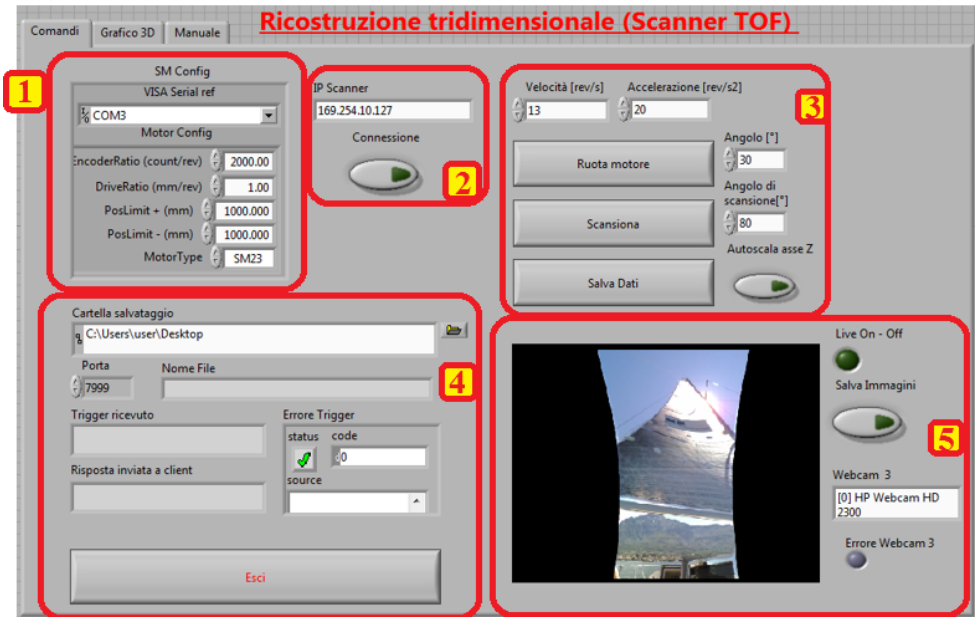


Figure 4.1: Acquisition software main screen

can be set (slow scans lead to a more dense point cloud, but for cases such as gennaker or spinnaker, flying shapes can be unstable and a quicker scan is suggested to retrieve a proper sail shape), scanning and start angle can be set too;

4. External trigger options: allows for receiving a command from another device to automatically run the scanning (useful for synchronous acquisition of different devices controlled by a common unit control)

5. Live visualization: displays the images acquired by a webcam mounted on top of the laser scanner. The webcam is useful to check the field covered by the scanner, so that the parameters of part 3 can be optimized. Moreover, it helps in understand possible unclear situations providing a visual support.

After the acquisition of the data, the point cloud are elaborated through another software developed in the contest of the present work, in Qt environment, using C++ programming language and including some open source libraries such as Point Cloud Library (PCL) and Computational Geometry Algorithms Library (CGAL). Figure 4.2 shows the graphical interface for the elaboration program.

The program is divided in two windows: the main window shown in the figure above and a tab widget where the user can set the parameters for each operation

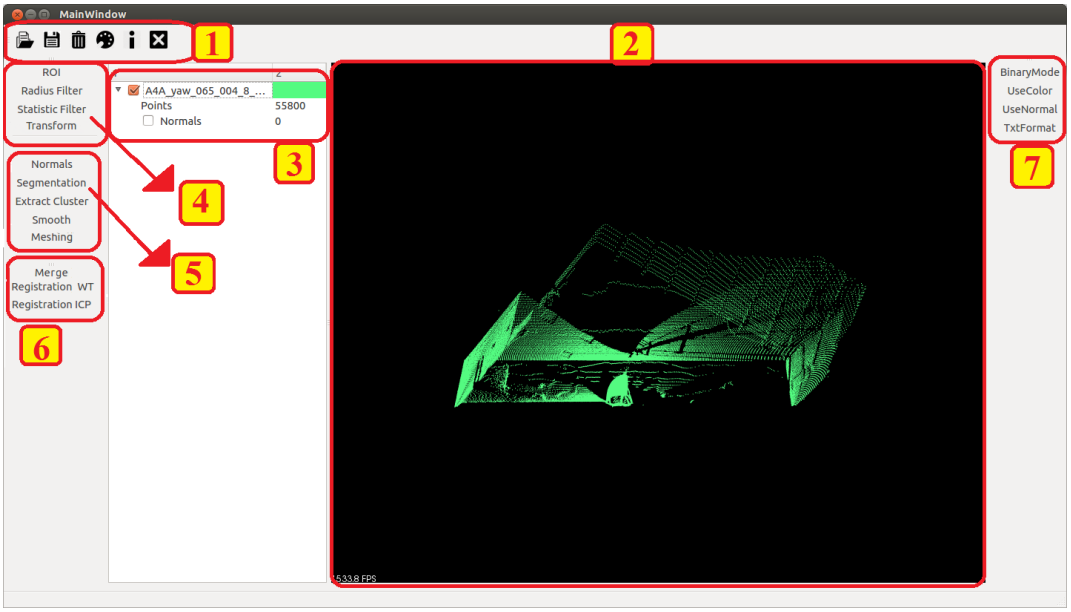


Figure 4.2: Post processing software main screen

implemented.

In the main window the user finds seven main blocks:

1. General operations: open file, save, delete, and exit are self-explanatory. The possibility of loading different types of file format has been implemented and thus the program can import Point Cloud Data (.pcd), Polygon File Format (.ply) and Text format (as the format of the acquisition output files). Color icon handles the color of the selected point cloud in the visualizer. Information icon opens a window where the available information (header of file, folder, number of points in the cloud etc..) are displayed.

2. Viewer: three dimensional viewer supplied by the PCL library. Point clouds, normals, triangulation can be visualized easily.

3. Treeview: a list of the loaded files is visualized. The name and the color of each point cloud is showed, additional lines can be showed trough a drop-down menu, where the number of samples and the normals are shown. The check button in the main line allows to hide or visualize the data set in the viewer, the same for normals associated to the data set for the secondary check button.

4. Transformations on points of a single cloud: outliers filtering (Spherical and

Statistical filters) and spatial filters (Region Of Interest) are available. Transformations usable are rotations around X, Y and Z axis and translations. Filters and rigid transformation parameters are adjustable in the secondary window tabs (see Figure 4.3a, Figure 4.3b and Figure 4.3c).

5. Single point cloud operations:

   (a) Normal Computation for the selected data set. Parameters for this operation are adjustable in the Normal tab (Figure 4.3d top).

   (b) Segmentation of the selected data set. Parameters for segmentation are set in the Segmentation tab (Figure 4.3e).

   (c) Cluster Extraction. Once segmented, the user can select a single cluster to extract and use as following data set.

   (d) Smoothing of the selected data set. Parameters for smoothing are available in the Smoothing tab (Figure 4.3d bottom).

   (e) Meshing of the selected data set. Parameters for the creation of the mesh are adjustable from the Mesh tab (Figure 4.3f).

6. Multiple point cloud operations, that requires at least two selected point clouds to be performed:

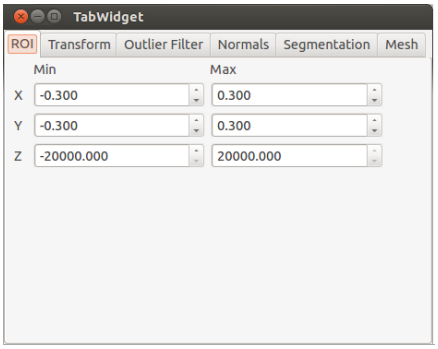
   (a) Merging of two or more selected point clouds. Merging of multiple data sets into a unique set. Selected clouds have to present the same properties.

   (b) Iterative Closets Point based Registration.

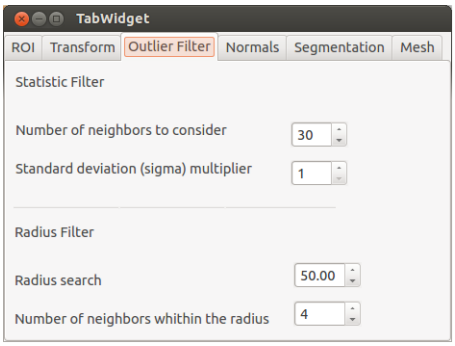
   (c) Point Clouds Registration for wind tunnel tests: a dedicated windows (see Figure 4.4) guides the user in the process of registering the clouds acquired in the wind tunnel environment (refer to section 4.4). The user has to fill the input fields requested, selecting a cloud in the dropdown menu to keep as a reference and the cloud to register with it, and picking three points in the reference cloud and three corresponding points in the other cloud. The first two points must belong to two different planar surfaces (for example to a wall and to the ceiling of the wind tunnel chamber) that will be used to calculate the rotation between the clouds; the last point must be a point visible in both cloud (for example the top

of the mast) and it will be used to compute the translation. When all the fields are filled, the user can start the registration procedure, clicking on the OK button and closing the registration window.

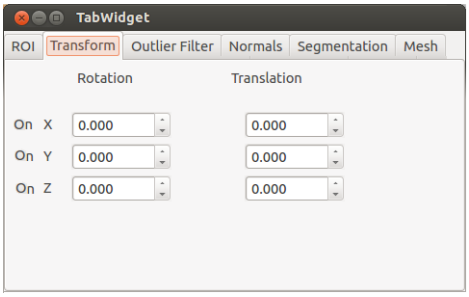
7. Saving Flags: helpful when the user needs to save a data set after having performed some operations. The standard saving function saves the cloud as a .pcd file, which is essentially an array of $n \times 3$ floating numbers, with additional headers. "Use color" and "Use normals" flags add color and normals information in the .pcd file, while "BinaryMode" save the data set in a binary file and "TxtFormat" in a .txt file instead of the .pcd file.



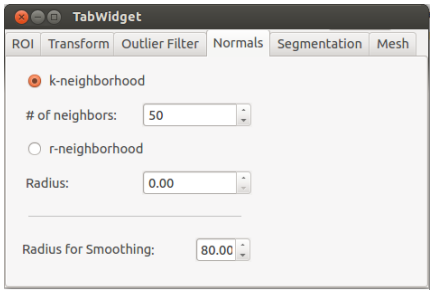
(a) ROI Filter Tab



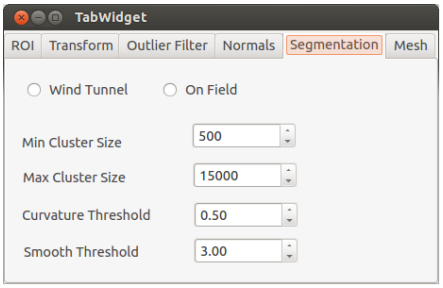
(b) Filters Tab



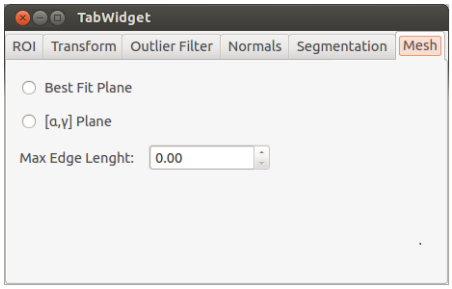
(c) Transform Tab



(d) Normals Tab



(e) Segmentation Tab



(f) Mesh Tab

Figure 4.3: Tab Widgets

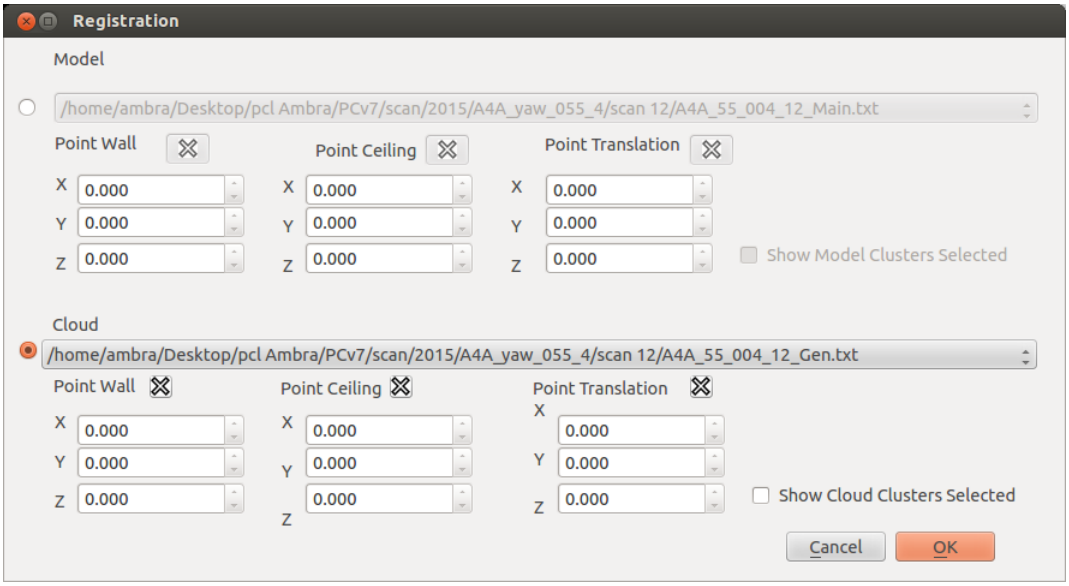


Figure 4.4: Wind tunnel registration window

4.2 Surface Reconstruction for Gennaker and Mainsail Acquired During Wind Tunnel Tests

In this paragraph, we report the elaboration results for some tests performed during a wind tunnel campaign aimed to support the sail inventory development for a high-performance superyacht. In particular, these tests helped in evaluating the opportunity to install on the boat a sail designed for furling. A complete 1:21 scale working yacht model was used, closely matching details of the real boat's rig and deck layout, allowing all the sails to be trimmed as in real life. The measurement of the overall wind loads on the hull and yacht sails is achieved using a six components force balance, which is placed inside the yacht hull. Over the course of a three-day wind tunnel campaign, five different gennaker sails were tested, in order to collect aerodynamic sail coefficients for different sail designs. Sails were trimmed to achieve maximum driving force by monitoring real-time force data while observing the sails directly from the control booth and using live video-feed from the cameras positioned in the wind tunnel. At each trim condition, 30 seconds of force data were recorded at 100Hz sample frequency and a scan was performed for both the mainsail and the gennaker simultaneously. Figure 4.5shows a photo taken during the tests for a so called A4A gennaker with apparent wind angle of 55°.

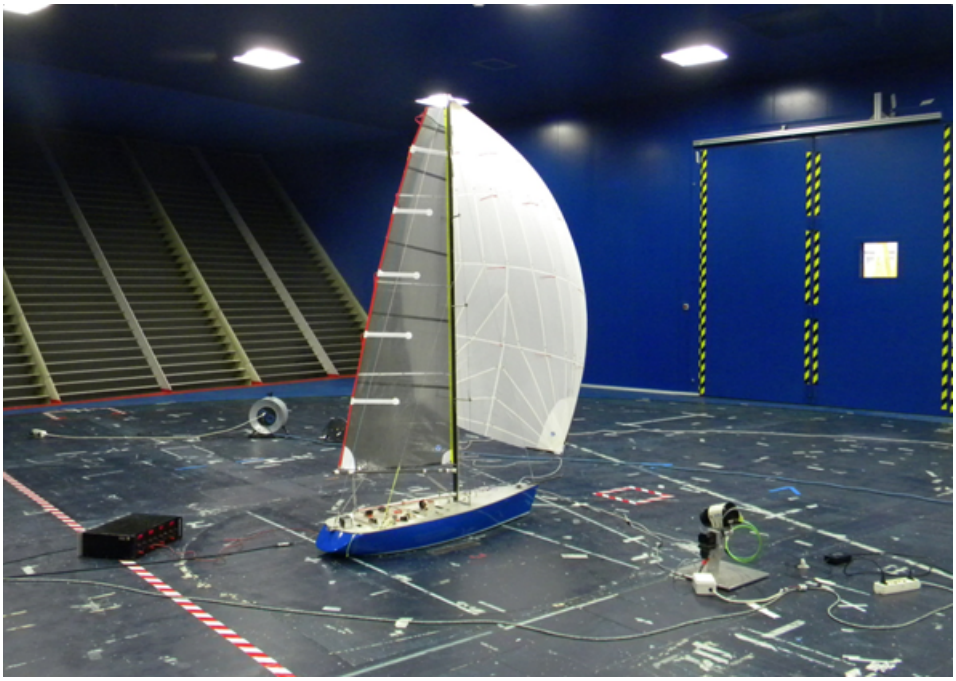
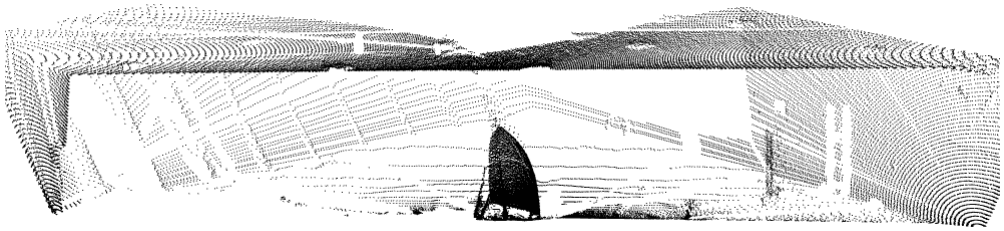
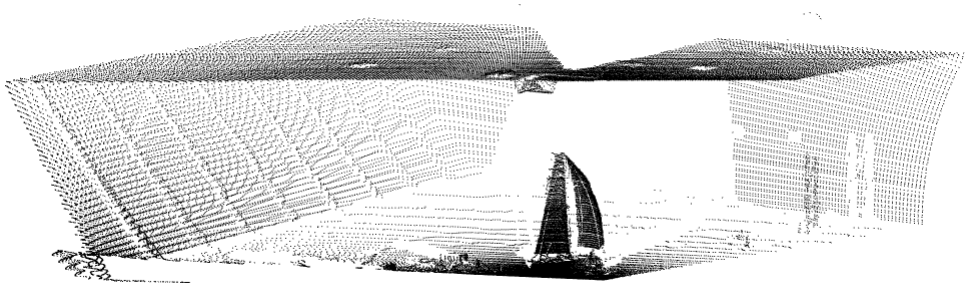


Figure 4.5: A4A sail prototype on the wind tunnel - only the gennaker dedicated acquisition unit is visible in the photo, the mainsail dedicated unit is more distant



(a) Gennaker dedicated acquisition



(b) Mainsail dedicated acquisition

Figure 4.6: Wind tunnel raw acquisitions

The acquisition unit dedicated to the gennaker scan is visible in front of the

yacht model; the one dedicated to the mainsail is far away. In fact, they have to be placed the more distant from the yacht as possible to avoid any flow perturbation and consequently any alteration of the sail shape; but at the same time the entire sail shape has to be acquired and possible occlusions have to be taken into account. Each acquisition unit is connected to a personal computer running the LABView acquisition data program. Acquisitions are triggered by a third computer, placed outside the wind tunnel chamber, which is connected to others via Ethernet.

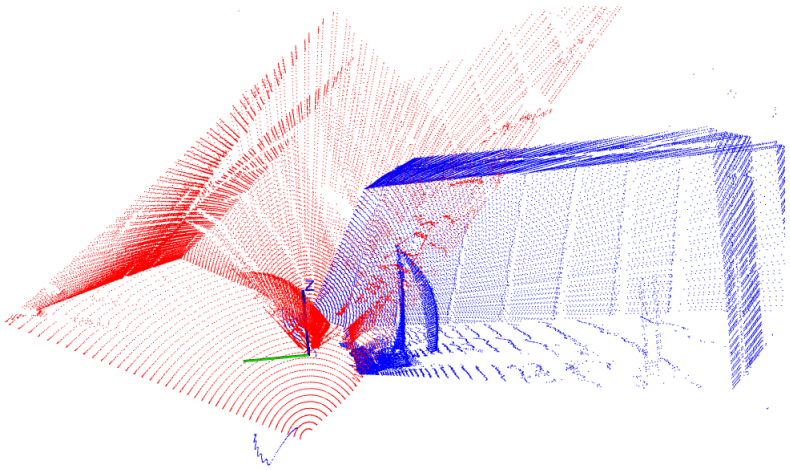


Figure 4.7: Wind tunnel raw acquisitions comparison - blue points correspond to the mainsail acquisition, red points to the gennaker acquisition

Parameters for scanning can be set differently for each unit, depending on the relative position with the yacht mode and on the sail scanned; however, when the trigger signal is launched, they start the acquisition simultaneously.

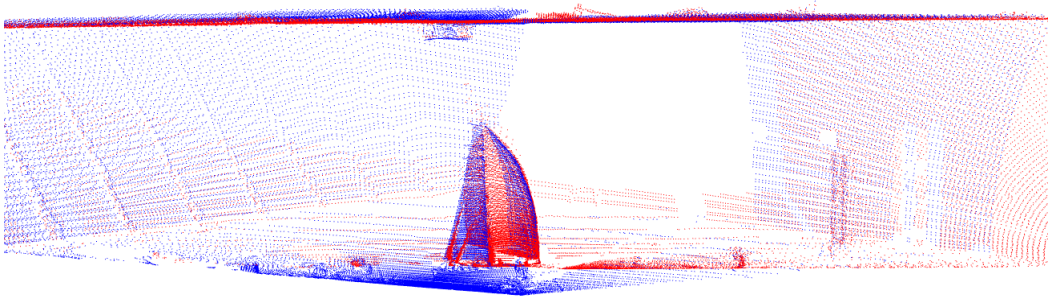


Figure 4.8: Point clouds after registration - blue points correspond to the mainsail acquisition, red points to the gennaker acquisition

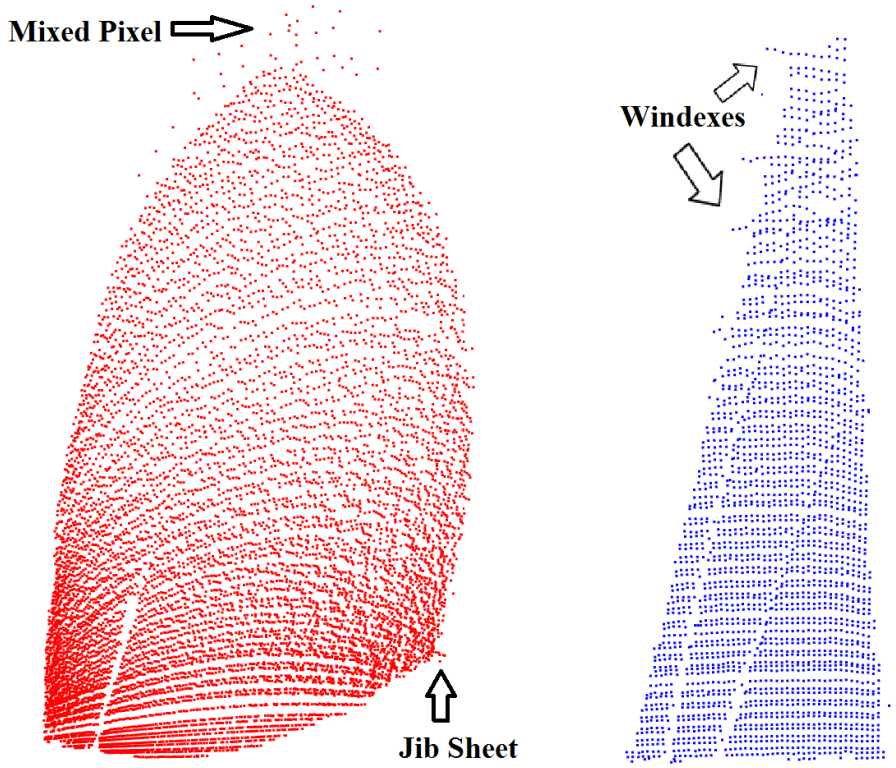
The output of a single test is then a couple of point cloud as the one presented

in Figure 4.6. Since the clouds are acquired from different acquisition units, placed in different spatial positions and under different scanning parameters, they do not match when overlapped as shown in Figure 4.7.

	c_{th}	θ_{th}	$\#_{neigh}$	N_{min}	N_{max}
Gennaker	0.9	7	50	5000	100000
Mainsail	0.8	7	50	5000	10000

Table 4.1: Case study segmentation parameters

To compose the whole yacht model they need to be registered together following the dedicated procedure explained in section 4.4. As output of the registration process the clouds can be merged to form a unique data set (see Figure 4.8). At this point, we need to extract the points belonging to the sail surfaces to reconstruct them.



(a) Gennaker (b) Mainsail

Figure 4.9: Point clouds after segmentation

We use the segmentation procedure implemented (refer to section 3.3.3) setting the threshold values reported in Table 4.1. Sail cluster is extracted from the whole

point cloud even if some points, clearly not belonging to the sail surface, remain. Issues behind them are various: points over the top of the gennaker are caused by mixed pixel problem (see section 2.4.3.8 for details) between the sail and the chamber ceiling; other points in the sail corners are due to the sail sheets; and finally sometimes points representing the windexes are enclosed (see Figure 4.9). These unwanted points presence are tricky to delete because there is no discontinuity between the sail surface and them, neither in terms of normals distribution. The cloud also presents a not uniform point density so a stricter segmentation would remove not only these samples but also samples from the actual shape. For the reconstruction purpose it is better to keep the most number of points belonging to the sail possible, so the choice is to use a more lenient segmentation and remove outliers via filtering or via point picking elimination. Point picking elimination consist in selecting a point in the viewer and removing them from the data set, this user intervention has to be introduced because in some dubious cases only the user can distinguish real outliers.

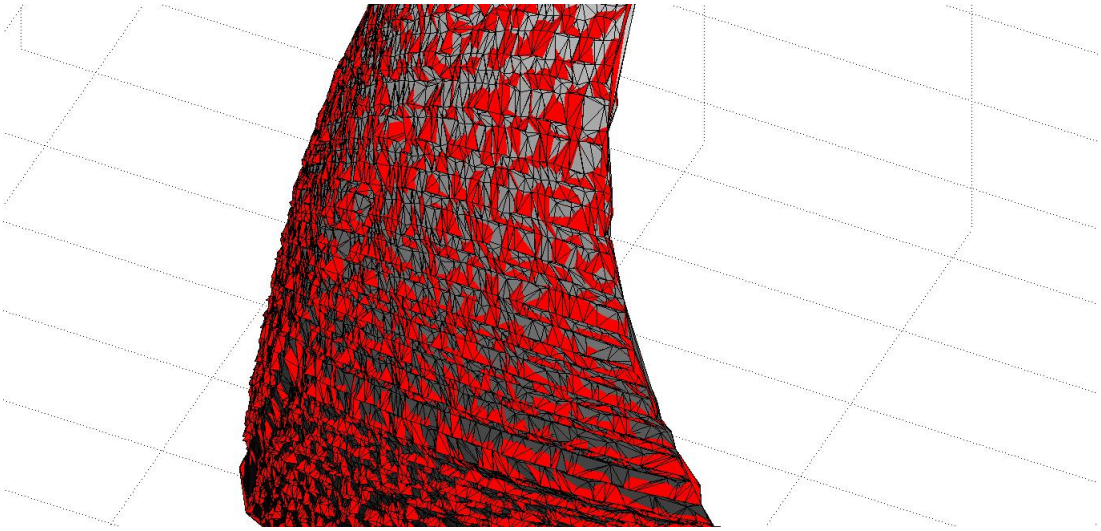
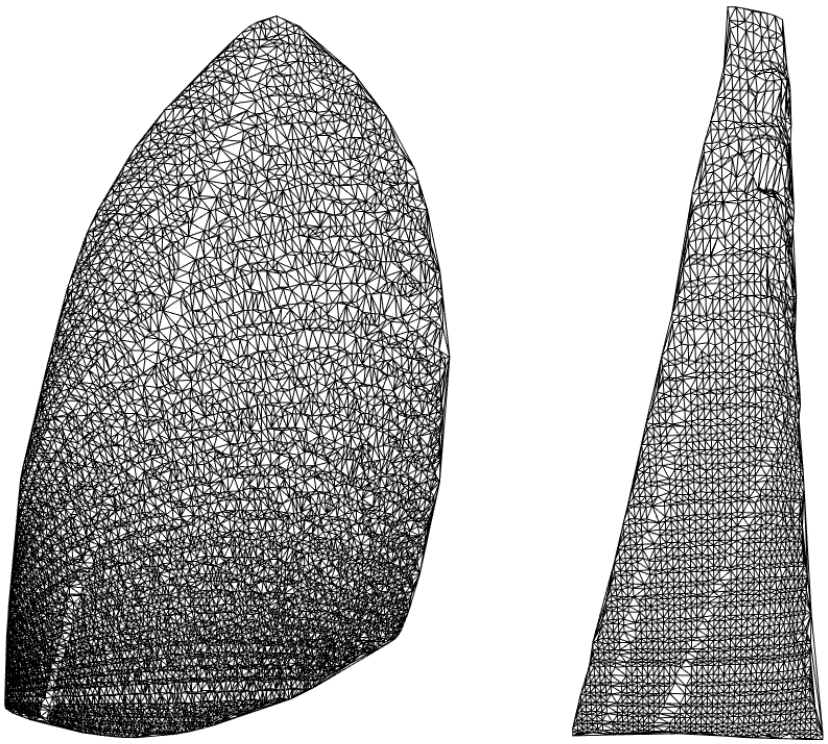


Figure 4.10: Comparison of mesh for raw (red) and resampled data (gray)

Then, this "clean" data sets are used for further operations. The surface of the sails could be already reconstructed from these point clouds, however it would result in an irregular and pointy mesh, due to the influence of acquisition noise. The data resampling step helps creating a smoother surface, as explained in section 3.4. Resampling is performed with a neighborhood size that limit the displacement of the points, such that the features of the sail are not modified and correct measurements

can be retrieved. Such value is identified in 75 mm for the gennaker and in 80 mm for the mainsail. The clouds after resampling are not shown, because the small changes are not visually appreciable while the differences appears after having computed the meshing step, as shown in Figure 4.10, where the red spikes provided by the mesh on the raw point cloud arise from the gray mesh realized from resampled data.

Finally, the cloud can be meshed, following the procedure developed in section 3.5.5. As said, triangulation can take place on the best fit plane or on the (α, γ) spherical coordinates plane at the user discretion, as visible from the Mesh tab, in Figure 4.3f.



(a) Gennaker (b) Mainsail

Figure 4.11: Sail Reconstruction

Results are similar in terms of mesh quality and the best choice can be selected only after the creation of both meshes and their comparison. For the A4A sail, in the considered case, the gennaker has been reconstructed on its best fit plane and it did not need longest triangle edge trimming, while the mainsail has been reconstructed on the (α, γ) plane and needed the edge trimming. The maximum edge length, is fixed at 150 mm, which is the value that eliminate all the unwanted

triangles. Figure 4.11 shows the above. After the reconstruction, the sail can be geometrically analyzed, or inserted on a CAD model of the yacht (see Figure 4.12). These CAD sails can be enclosed into a CFD codes and yacht performances can be assessed.

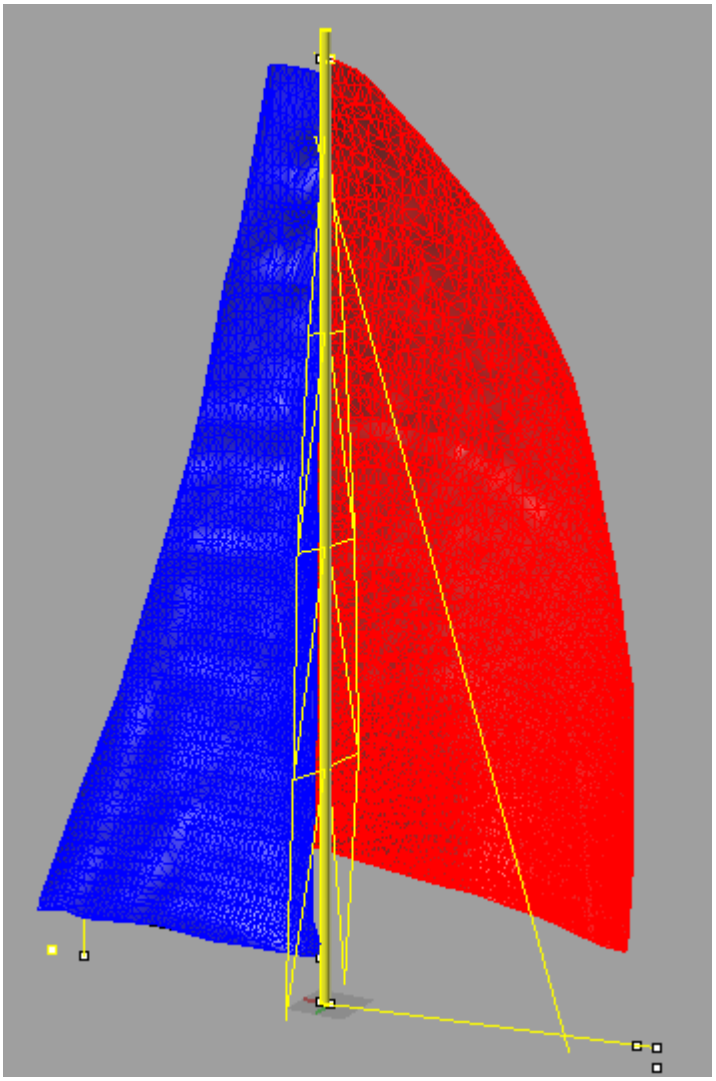


Figure 4.12: Final model of the sails

4.3 Surface Reconstruction for Jib and Mainsail Acquired During On Field Tests

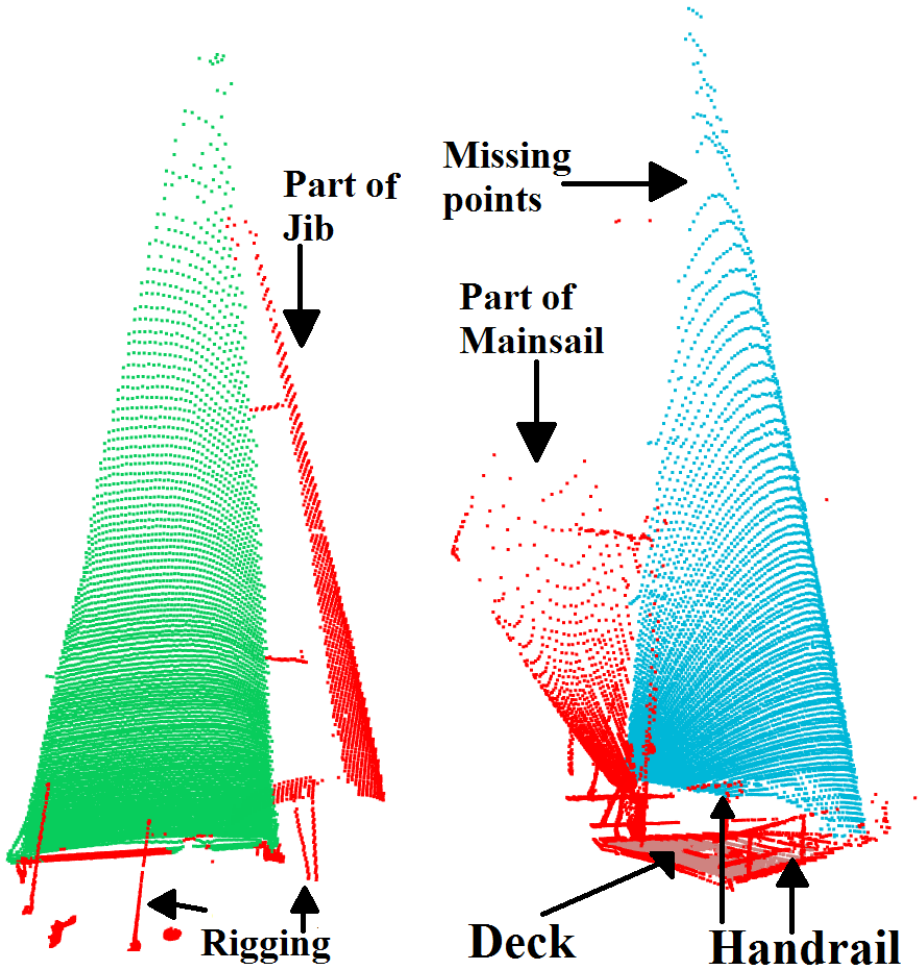
In this paragraph, we report the results of the elaboration of some point clouds acquired on the Laboratory Boat of the Lecco Innovation Hub at Colico. The Laboratory Boat (a 35 meters long Comet) is designed to function as a dynamometric scale that can acquire previously unavailable data on the aerodynamic and hydrodynamic loads acting on the main components of the yacht. The heart of the system is a framework inside the hull that allows the entire rig and sail plan to be connected to a system of load cells to measure the overall forces and moments transmitted by the sail plan to the boat when under sail. For these tests an acquisition unit was placed at the bottom of the mast (12 meters high) and scanned the jib while another unit was placed next to the winch to scan the mainsail. Figure 4.13 presents a model of the boat and the positions of the acquisition units are marked by the red stars.

(a) Mainsail dedicated acquisition device (b) Jib dedicated acquisition device

Figure 4.13: The Laboratory Boat (LIH)

The software described above elaborated the data. Elaboration parameters have to be adapted to the specific case and the registration step is performed using the common algorithm available in literature (refer to section 3.2.1), selecting correspondent points such as the top of the mast and the crosstree ends. Differently from the wind tunnel acquisitions, there is no background, so the custom registration procedure, developed in section 4.4, is ineffective. To overcome this deficit, in future on field campaign, the presence of two planar panels will be tested to be

used as reference for registration. The panels should be large enough, and placed such as, both laser scanner see them, but without interfering with the wind flow. Then, the interpretation of the data set starts.



(a) Mainsail (b) Jib

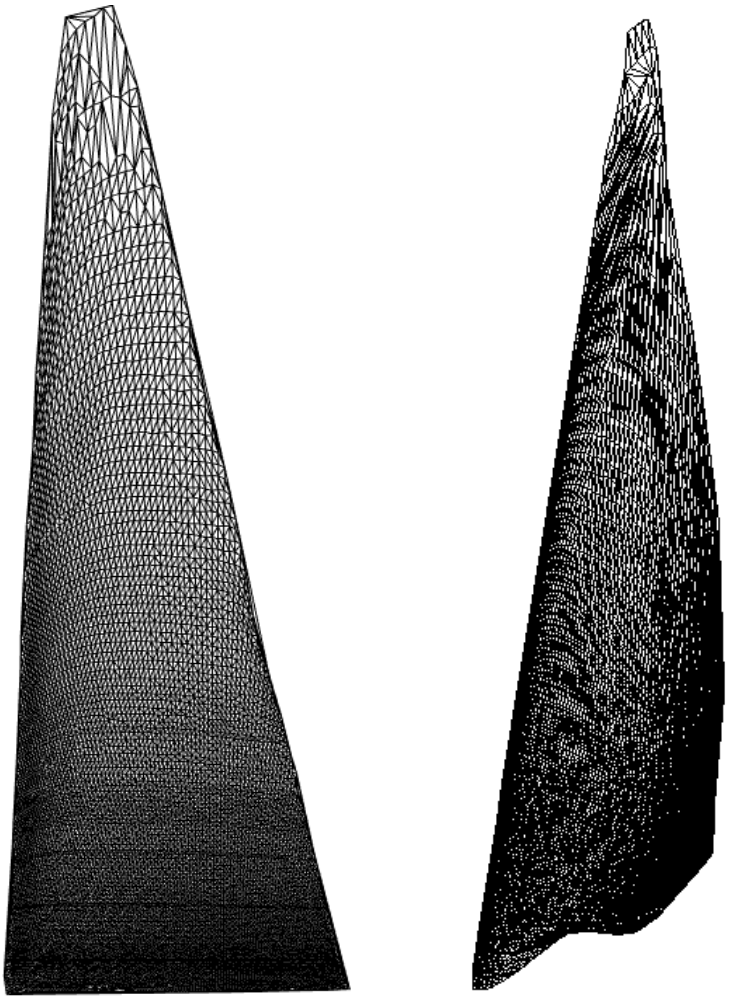
Figure 4.14: Segmentation for on field acquisitions

	c_{th}	θ_{th}	$\#_{neigh}$	N_{min}	N_{max}
Jib	1	10	20	5000	100000
Mainsail	1.5	4	30	6000	100000

Table 4.2: On field acquisitions segmentation parameters

Point clouds representing on filed acquisition are much more complex then wind tunnel scenes. Many objects needed on a real yacht are not present on the yacht

model used in the wind tunnel, such as the handrails (see Figure 4.13), many shrouds, etc.. Moreover, environmental conditions during navigation can't be controlled as in the wind tunnel, leading to difficulties in acquiring a correct shape for the considered trim. The acquisitions used in this chapter correspond to a close-hauled trim, this means that the wind direction is close to the bow-stern direction, coming from forward.



(a) Mainsail (b) Jib

Figure 4.15: Surfaces reconstructed for on field acquisitions

With this trim, the bow sail used is the jib instead of the gennaker. The segmented data sets are presented in Figure 4.14 and the parameters used are displayed in Table 4.2. Photo in Figure4.13 and Figure4.14.b shows that during the acquisition the jib leans on the handrail, moreover on the top, some points are missed, due

to problems reported in section 2.4.3; this is an example of possible problems that can be encountered during on field acquisitions. On the other hand, Figure 4.14.a shows difficulties in the segmentation, as some points belonging to the boom can't be separated from the mainsail. Stricter values for the segmentation parameters do not improve the overall quality of the interpretation, since they would lead to the exclusion of points that actually belong to the surface, on top of the mainsail. The smoothing is applied, with neighbors size equal to $h = 150$ mm for the mainsail and $h = 275$ mm for the jib. Due to the high jib twisting the best fitting plane triangulation is not applicable because in the projection some points would overlap. This would cause the creation of a wrong 3D mesh. The main problem of the jib data set is however the loss of points in the top part of the sail. This issue can't be fixed with the post processing techniques presented in this thesis, in fact the final mesh, although generically good, does not represent a smooth surface at the top, as visible in Figure 4.15.b.

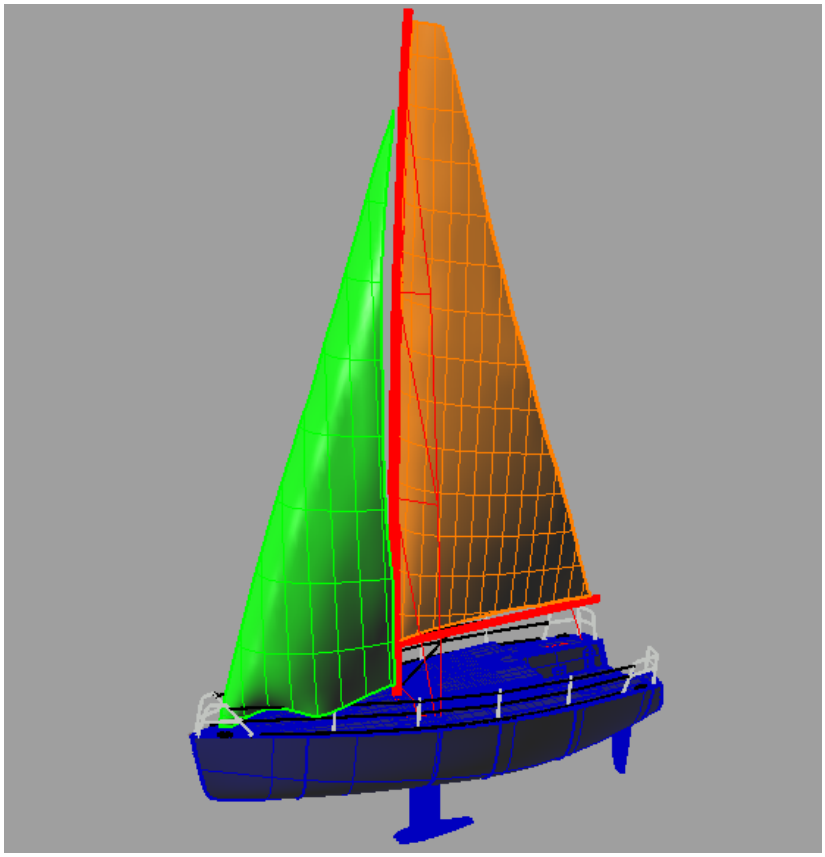


Figure 4.16: Jib and Mainsail reconstructed and placed onto the yacht CAD model

The visualization of the jib in Figure 4.15.b is rotated by 180 degrees on its vertical axes, respect to Figure 4.14.b (i.e. view from outside the yacht instead of from inside the yacht), to allow a better view of the reconstruction of the sail leant on the handrail. The good surface reconstruction of this problem demonstrates that the main problem of the procedure is the loss of points rather than a bad acquisition due to environmental condition. On the other hand, the mainsail is reconstructed with good accuracy, the surface showed in Figure 4.15.a is created with the projection of the points on the spherical coordinates plane, trimming edges longer than 400 mm. Once more, the sail surface reconstructed are reported onto the CAD model of the yacht and Figure 4.16 presents the final output.

Chapter 5

Conclusions

The presented work presents an algorithm for the 3D reconstruction of the flying sail shape acquired by means of a home made innovative device. The main goal is to capture the sail shape in a specific moment during navigation and to use it, together with force and moments values, as an input for more reliable CFD computation or as a reference for fluid structure interaction codes. Different studies are reported in literature regarding the acquisition of the flying shapes. Most of them are based on photogrammetry technique that allows for retrieving the geometry of some colored stripes placed on the sail surface. This technique requires at least a pair of camera for each sail and assumes that the stripes remain in a horizontal plane which can be questionable especially for downwind sails. Thus, after a brief analysis of different non-contact techniques, reported in chapter 2, we chose to exploit the Time Of Flight methodology and we realized a specific acquisition unit (currently patent pending) enclosing a SICK LMS 511 laser scanner. This device allows us to obtain 3D measurements of the entire sail surface, no matter the sail type neither the flying shape assumed. This kind of sensors were originally design for security application, thus a detailed metrological qualification was necessary to assess the unit measurements error. We analyzed the performances only in the range of interest, as to say one meter up to 15 meters. We could observe that the systematic error rises with the distance, but remains an acceptable value for our application, in fact, the maximum percentage error retrieved is 0.27% of the acquired distance. Moreover, the statistical error remains almost constant, around the value of 4 mm, within the analyzed range. The presence of a critical incident angle between the laser ray direction and the normal to the target surface was worked out: incident angles greater than 70° lead to a significant drop in the

measurement accuracy. Outdoor tests were performed highlighting the influence of the ambient light: measurement standard deviation can increase by the 350% while the sun hits directly the sensor. In addition, different target materials were tested, proving that the sail tissue presents the widest uncertainty due to its its transparent nature; and other tests, on the influence of the target color, suggested the application of a light gray colored film on the the sail. This expedient has still to be investigate trying to evaluate whether the addition of material might alteration the flying shape of the sail, due to the load effect. Other details about the metrological qualification work are reported in section 2.4.3 that is part of the paper "Characterization of a 2-D Laser Scanner for Outdoor Wide Range Measurement", presented at the 2014 AIVELA conference. Abstract is reported in Appendix 6.

Once the raw data are acquired, they are stored as point cloud structure and consequently elaborated as described in Chapter 3. A dedicated software was entirely developed using all open source libraries such as PCL or CGAL, programming in C++ language, in Qt environment on a Linux based computer. Each step of the post processing procedure is discussed and examples of the elaboration on synthetic data and on real acquisitions are reported to justify the algorithm developed. First of all, we describe the registration step. Each yacht sail is scanned by a different acquisition unit. This leads to the need for registering different scans in a common coordinate system, in order to compose an unique rig for the same yacht. To perform this operation a specific algorithm has been developed, which relies on information taken from the acquired overall scene. For wind tunnel acquisitions, we decided to align all the planar surface easily recognizable such as walls and ceiling. Registration of on field acquisition is more tricky, since the two scanners acquire very few common points. In future campaign the use of planar panels, placed such as they are visible from both devices, will be tested.

Then, registered scenes have to be interpreted, that means recognizing different objects in the point cloud and organizing them in distinct clusters. Our final goal is the extraction of the point cluster that represents the sail surface. Proximity to other points and normal trends are exploited as discriminant criteria. Unwanted points might remain after the segmentation step due to objects close to the sail surface, such as the boom, shrouds and spreaders or due to errors in the acquisition process, like mixed pixels. The sail cluster is then filtered to face this problem. Sometimes user intervention is required. Lats step is the 3D surface reconstruction.

Surface retrieving from segmented data presents a pointy surface, due to the

presence of acquisition noise. To reduce its influence a resampling step is applied before reconstructing the sail, whose main result is the generation of more smooth surface. Resampling is substantially a local interpolation of the data. A heavy smoothing would alter the overall sail shape, which is obviously unfair. Evaluating this resampling effect is not an easy task, as the threshold for smoothing should be defined by displacements from a reference surface, which is not known as it is the final object searched in this thesis. For the current work, the smoothness operation is fixed as for avoiding loss of samples during the implementation. A custom algorithm was developed. In particular, the sail reconstruction step applies a change from the 3D space, native of the data, to their 2D projection, where a mesh is computed. The two dimensional mesh is then reprojected in the [x,y,z] space, as the 2D points and 3D points are uniquely linked. Good results were obtained (according to the sail makers' feedback) and a good level of automation was achieved. However, several improvements can be undertaken.

5.1 Future work

As said before, some improvements proposed for the acquisition phase has yet to be tested. Use of adhesive plastic films on the sail tissue and the presence of rigid panels on the scenes acquired during on-field campaign, has been proposed respectively to avoid possible data loss and to achieve a more robust cloud registration. This expedients not only have to prove to help the acquisition accuracy, but, at the same time, they have to prove not to influence the original shape of the sail.

On the software side, efforts have to be made to improve the robustness of the program, and to reduce user interaction. The data resampling progress could be even more investigated, focusing on the possibility of accepting downsampling of the point cloud, if it does not alter the overall shape of the sail, neither influence its geometric parameters. Moreover, during the choice of the algorithm for the reconstruction of the sail surface, some approximating algorithm has been considered, rather than Delaunay triangulation, which interpolate the samples. This procedures were the Poisson surface reconstruction and the B-spline surface reconstruction. Both generated surfaces with a better overall quality, but they also exceeded the border of the actual shape to retrieve. A further study on the properties of approximated surfaces should be of interest for the further developments of the considered work. The B-spline method was particularly promising since its

implementation (through the PCL library) came along with a function to retrieve the correct edges of the surface. This function was able to reconstruct the gennaker edges, but failed in the mainsail case. A similar, custom function, could be developed to work in the specific context of sail reconstruction.

Many small improvements can still be added to the software, like the possibility to save the meshes generated in different file formats. The software remains a work in progress also because the PCL library, which is the main part of the developed program, continues to evolve, repair errors and add functionalities. Even during the development of the software application, some bugs were reported to the PCL community. Fixing of this errors opens new ways to handle the post-processing operations, simplifying even more the role of the external user.

Chapter 6

Paper

Characterization of a 2-D laser scanner for outdoor wide range measurement

Department of Mechanical Engineering, Politecnico di Milano, via La Masa 1, 20156 Milano, Italy

Eugenio Canciani Ambra Vandone Remo Sala

Email: eugenio.canciani@mail.polimi.it *Email: ambra.vandone@polimi.it* *Email: remo.sala@polimi.it*

Abstract:

This paper presents a metrological characterization study of SICK LMS 511 laser scanner, with an extended analysis of its main acquisition issues. Various parameters that could affect the sensor performances, such as warm-up time, target properties (color and material), and target position (distance and orientation) are investigated. Moreover, the mixed pixel problem is introduced and, finally, since the sensor is designed to work in a wide outdoor environment, the effect of direct sun light is taken into account. Some cases of faulty data are identified and explanations discussed.

Bibliography

[Adams and Bischof, 1994] Adams, R. and Bischof, L. (1994). Seeded region growing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(6):641–647.

[Akkiraju et al., 1995] Akkiraju, N., Edelsbrunner, H., Facello, M., Fu, P., Mücke, E., and Varela, C. (1995). Alpha shapes: definition and software. In *Proceedings of the 1st International Computational Geometry Software Workshop*, pages 63–66.

[Alexa et al., 2003] Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., and Silva, C. T. (2003). Computing and rendering point set surfaces. *Visualization and Computer Graphics, IEEE Transactions on*, 9(1):3–15.

[Alwan et al., 2005] Alwan, M., Wagner, M. B., Wasson, G., and Sheth, P. (2005). Characterization of infrared range-finder pbs-03jn for 2-d mapping. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3936–3941. IEEE.

[Arun et al., 1987] Arun, K. S., Huang, T. S., and Blostein, S. D. (1987). Least-squares fitting of two 3-d point sets. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (5):698–700.

[Aurenhammer, 1991] Aurenhammer, F. (1991). Voronoi diagramsâĂŤa survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405.

[Bendat and Piersol, 2011] Bendat, J. S. and Piersol, A. G. (2011). *Random data: analysis and measurement procedures*, volume 729. John Wiley & Sons.

[Besl and McKay, 1992] Besl, P. J. and McKay, N. D. (1992). Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics.

[Bowyer, 1981] Bowyer, A. (1981). Computing dirichlet tessellations. *The Computer Journal*, 24(2):162–166.

[Carlson, 2003] Carlson, W. (2003). A critical history of computer graphics and animation. *The Ohio State University*.

[Chen and Medioni, 1991] Chen, Y. and Medioni, G. (1991). Object modeling by registration of multiple range images. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 2724–2729. IEEE.

[Clauss and Heisen, 2000] Clauss, G. and Heisen, W. (2000). Cfd analysis on the flying shape of modern yacht sails. *The Quest for the Origins of Life*, page 87.

[Cox, 1972] Cox, M. G. (1972). The numerical evaluation of b-splines. *IMA Journal of Applied Mathematics*, 10(2):134–149.

[Curless, 1999] Curless, B. (1999). Overview of active vision techniques. *SIGGRAPH 99 Course on 3D Photography*.

[D'Acquisto et al., 2002] D'Acquisto, L., Fratini, L., and Siddiolo, A. (2002). A modified moiré technique for three-dimensional surface topography. *Measurement Science and Technology*, 13(4):613.

[De Berg et al., 2000] De Berg, M., Van Kreveld, M., Overmars, M., and Schwarzkopf, O. C. (2000). *Computational geometry*. Springer.

[De Boor, 1972] De Boor, C. (1972). On calculating with b-splines. *Journal of Approximation Theory*, 6(1):50–62.

[Declerck et al., 1997] Declerck, J., Feldmar, J., Goris, M. L., and Betting, F. (1997). Automatic registration and alignment on a template of cardiac stress and rest reoriented spect images. *Medical Imaging, IEEE Transactions on*, 16(6):727–737.

[DeFanti and Brown, 1991] DeFanti, T. A. and Brown, M. D. (1991). Visualization in scientific computing. *Advances in Computers*, 33(1):247–305.

[Delaunay, 1934] Delaunay, B. (1934). Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2.

[Deparday et al.,] Deparday, J., Bot, P., Hauville, F., Motta, D., Le Pelley, D., and Flay, R. Dynamic measurements of pressures, sail shape and forces on a full-scale spinnaker.

[Duda and Hart, 1972] Duda, R. O. and Hart, P. E. (1972). Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15.

[Dursun et al., 2003] Dursun, A., Ecevït, N., and Özder, S. (2003). Application of wavelet and fourier transforms for the determination of phase and three-dimensional profile. In *Proceedings of International Conference on Signal Processing, Int. J. Comput. Intell*, volume 1, pages 168–172.

[Edelsbrunner and Mücke, 1994] Edelsbrunner, H. and Mücke, E. P. (1994). Three-dimensional alpha shapes. *ACM Transactions on Graphics (TOG)*, 13(1):43–72.

[Eggert et al., 1998] Eggert, D. W., Fitzgibbon, A. W., and Fisher, R. B. (1998). Simultaneous registration of multiple range views for use in reverse engineering of cad models. *Computer Vision and Image Understanding*, 69(3):253–272.

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

[Fortune, 1987] Fortune, S. (1987). A sweepline algorithm for voronoi diagrams. *Algorithmica*, 2(1-4):153–174.

[Fortune, 1992] Fortune, S. (1992). Voronoi diagrams and delaunay triangulations. *Computing in Euclidean geometry*, 1:193–233.

[Fossati et al., 2008] Fossati, F., Martina, F., and Muggiasca, S. (2008). Experimental database of sail performance and flying shapes in upwind conditions. In *Proc. International Conference on Innovation in High Performance Sailing Yachts, Lorient, France, May*, pages 29–30.

[Fossati et al., 2013] Fossati, F., Muggiasca, S., Bayati, I., and Bertorello, C. (2013). Lecco innovation hub sailing yacht lab project–a sailing research infrastructure. In *Proceedings of the 3rd International Conference on Innovation in High Performance Sailing Yachts, Lorient, France*, pages 255–260.

[Frey and Borouchaki, 1999] Frey, P. J. and Borouchaki, H. (1999). Surface mesh quality evaluation. *International journal for numerical methods in engineering*, 45(1):101–118.

[Gopi and Krishnan, 2002] Gopi, M. and Krishnan, S. (2002). A fast and efficient projection-based approach for surface reconstruction. In *Computer Graphics and Image Processing, 2002. Proceedings. XV Brazilian Symposium on*, pages 179–186. IEEE.

[Graf and Müller, 2009] Graf, K. and Müller, O. (2009). Photogrammetric investigation of the flying shape of spinnakers in a twisted flow wind tunnel. In *The 19th Chesapeake Sailing Yacht Symposium, Annapolis, MD*.

[Guibas and Stolfi, 1985] Guibas, L. and Stolfi, J. (1985). Primitives for the manipulation of general subdivisions and the computation of voronoi. *ACM Transactions on Graphics (TOG)*, 4(2):74–123.

[Guidi et al., 2010] Guidi, G., Russo, M., and Beraldin, J.-A. (2010). *Acquisizione 3D e modellazione poligonale*. McGraw-Hill.

[Hajeer et al., 2001] Hajeer, M. Y., Ayoub, A. F., Millett, D. T., Bock, M., and Siebert, J. (2001). Three-dimensional imaging in orthognathic surgery: the clinical application of a new method. *The International journal of adult orthodontics and orthognathic surgery*, 17(4):318–330.

[Harthong et al., 1991] Harthong, J., Sahli, H., Poinsignon, R., and Meyrueis, P. (1991). Analyse de formes par moiré. *Journal de Physique III*, 1(1):69–84.

[Hochkirch and Brandt, 1999] Hochkirch, K. and Brandt, H. (1999). Full-scale hydrodynamic force measurement on the berlin sailing dynamometer. In *Proc. 14th Chesapeake Sailing Yacht Symposium, Annapolis, MD, Jan*, volume 30, pages 33–44.

[Hough, 1962] Hough, P. V. (1962). Method and means for recognizing complex patterns. Technical report.

[Huang et al., 2004] Huang, W., Kong, Z., Ceglarek, D., and Brahmst, E. (2004). The analysis of feature-based measurement error in coordinate metrology. *IIE Transactions*, 36(3):237–251.

[Johnson and Kang, 1999] Johnson, A. E. and Kang, S. B. (1999). Registration and integration of textured 3d data. *Image and vision computing*, 17(2):135–147.

[Kazhdan et al., 2006] Kazhdan, M., Bolitho, M., and Hoppe, H. (2006). Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7.

[Kneip et al., 2009] Kneip, L., Tâche, F., Caprari, G., and Siegwart, R. (2009). Characterization of the compact hokuyo urg-04lx 2d laser range scanner. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1447–1454. IEEE.

[Knopp et al., 2010] Knopp, J., Prasad, M., Willems, G., Timofte, R., and Van Gool, L. (2010). Hough transform and 3d surf for robust three dimensional classification. In *Computer Vision–ECCV 2010*, pages 589–602. Springer.

[Kobayashi et al., 1990] Kobayashi, T., Ueda, K., Honma, K., Sasakura, H., Hanada, K., and Nakajima, T. (1990). Three-dimensional analysis of facial morphology before and after orthognathic surgery. *Journal of Cranio-Maxillofacial Surgery*, 18(2):68–73.

[Le Pelley and Modral, 2008] Le Pelley, D. and Modral, O. (2008). V-spars: A combined sail and rig shape recognition system using imaging techniques. In *Proc. 3rd High Performance Yacht Design Conference Auckland, New Zealand, Dec*, pages 2–4.

[Lee and Ehsani, 2008] Lee, K.-H. and Ehsani, R. (2008). Comparison of two 2d laser scanners for sensing object distances, shapes, and surface patterns. *Computers and electronics in agriculture*, 60(2):250–262.

[Levin, 2004] Levin, D. (2004). Mesh-independent surface interpolation. In *Geometric modeling for scientific visualization*, pages 37–49. Springer.

[Marshall et al., 1998] Marshall, S. J., Rixon, R. C., Whiteford, D. N., and Cumming, J. T. (1998). The orthoform 3-dimensional clinical facial imaging system. In *Proceedings of the 15th IFHE Congress*, volume 15, pages 83–7. Citeseer.

[Marton et al., 2009] Marton, Z. C., Rusu, R. B., and Beetz, M. (2009). On fast surface reconstruction methods for large and noisy point clouds. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3218–3223. IEEE.

[Masuyama and Fukasawa, 1997] Masuyama, Y. and Fukasawa, T. (1997). Full-scale measurements of sail force and validation of numerical calculation method. In *Proc. 13th Chesapeake Sailing Yacht Symposium, Annapolis, MD, Jan*, volume 25, pages 23–36.

[Masuyama et al., 2009] Masuyama, Y., Tahara, Y., Fukasawa, T., and Maeda, N. (2009). Database of sail shapes versus sail performance and validation of numerical calculations for the upwind condition. *Journal of marine science and technology*, 14(2):137–160.

[Mencl and Muller, 1997] Mencl, R. and Muller, H. (1997). Interpolation and approximation of surfaces from three-dimensional scattered data points. In *Scientific Visualization Conference, 1997*, pages 223–223. IEEE.

[Molton et al., 1998] Molton, N., Se, S., Brady, J., Lee, D., and Probert, P. (1998). A stereo vision-based aid for the visually impaired. *Image and vision computing*, 16(4):251–263.

[Motta et al., 2014] Motta, D., Flay, R., Richards, P., Le Pelley, D., Deparday, J., and Bot, P. (2014). Experimental investigation of asymmetric spinnaker aerodynamics using pressure and sail shape measurements. *Ocean Engineering*, 90:104–118.

[Mücke, 1993] Mücke, E. P. (1993). Shapes and implementations in three-dimensional geometry.

[Park and Subbarao, 2003] Park, S.-Y. and Subbarao, M. (2003). An accurate and fast point-to-plane registration technique. *Pattern Recognition Letters*, 24(16):2967–2976.

[Patorski, 1993] Patorski, K. (1993). *Handbook of the moiré fringe technique*. Elsevier Science.

[Perego, 2008] Perego, A. (2008). *Qualificazione metrologica di uno scanner laser a tempo di volo per la ricostruzione tridimensionale di fomre in ambiente non strutturato*. Politecnico di Milano.

[Piegl and Tiller, 1996] Piegl, L. A. and Tiller, W. (1996). The nurbs book (monographs in visual communication).

[Pottmann et al., 2004] Pottmann, H., Leopoldseder, S., and Hofer, M. (2004). Registration without icp. *Computer Vision and Image Understanding*, 95(1):54–71.

[Pottmann and Wallner, 2009] Pottmann, H. and Wallner, J. (2009). *Computational line geometry*. Springer Science & Business Media.

[Ranzenbach R., 2002] Ranzenbach R., K. J. (2002). Utility of flying shapes in the development of offwind sail design databases. *2nd High performance Yacht Design Conference*.

[Reina and Gonzales, 1997] Reina, A. and Gonzales, J. (1997). Characterization of a radial laser scanner for mobile robot navigation. In *Intelligent Robots and Systems, 1997. IROS'97., Proceedings of the 1997 IEEE/RSJ International Conference on*, volume 2, pages 579–585. IEEE.

[Renzsch and Graf, 2011] Renzsch, H. and Graf, K. (2011). An experimental validation case for fluid-structure-interaction simulations of downwind sails. In *Proceedings of the 21th Chesapeake Sailing Yacht Symposium*.

[Ristic and Brujic, 1997] Ristic, M. and Brujic, D. (1997). Efficient registration of nurbs geometry. *Image and Vision Computing*, 15(12):925–935.

[Rusu, 2010] Rusu, R. B. (2010). Semantic 3d object maps for everyday manipulation in human living environments. *KI-Künstliche Intelligenz*, 24(4):345–348.

[Rusu et al., 2009] Rusu, R. B., Blodow, N., and Beetz, M. (2009). Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE.

[Salzmann et al., 2007] Salzmann, M., Pilet, J., Ilic, S., and Fua, P. (2007). Surface deformation models for nonrigid 3d shape recovery. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(8):1481–1487.

[Schnabel et al., 2006] Schnabel, R., Wahl, R., and Klein, R. (2006). Shape detection in point clouds. *Computer Graphics Technical Reports*, 2:2.

[Seidel, 1988] Seidel, R. (1988). Constrained delaunay triangulations and voronoi diagrams with obstacles. *Rep*, 260:178–191.

[Sheet, 2014] Sheet, S. D. (2014). SICK Data Sheet. `https://www.mysick.com/eCat.aspx?go=DataSheet&Cat=Row&At=Fa&Cult=English&ProductID=45448&Category=Produktfinder./`. [ ].

[Sloan, 1993] Sloan, S. (1993). A fast algorithm for generating constrained delaunay triangulations. *Computers & Structures*, 47(3):441–450.

[Soderkvist, 2014] Soderkvist, I. (2014). Using svd for some fitting problems.

[Takeda and Mutoh, 1983] Takeda, M. and Mutoh, K. (1983). Fourier transform profilometry for the automatic measurement of 3-d object shapes. *Applied optics*, 22(24):3977–3982.

[Trucco and Verri, 1998] Trucco, E. and Verri, A. (1998). *Introductory techniques for 3-D computer vision*, volume 201. Prentice Hall Englewood Cliffs.

[Watson, 1981] Watson, D. F. (1981). Computing the n-dimensional delaunay tessellation with application to voronoi polytopes. *The computer journal*, 24(2):167–172.

[Ye and Borenstein, 2002] Ye, C. and Borenstein, J. (2002). Characterization of a 2-d laser scanner for mobile robot obstacle negotiation. In *ICRA*, pages 2512–2518.

[Zhang, 1992] Zhang, Z. (1992). Iterative point matching for registration of free-form curves.