TÉCNICO LISBOA

Improving image stitching methods for mapping low texture wind turbine blades

Inês Gaspar Coelho

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Alexandre José Malheiro Bernardino
Prof. Manuel Fernando Cabido Peres Lopes

Examination Committee

Chairperson: Prof. João Manuel de Freitas Xavier
Supervisor: Prof. Alexandre José Malheiro Bernardino
Member of the Committee: Prof. João Paulo Salgado Arriscado Costeira

June 2022

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

Thank you to my parents for their friendship, encouragement and caring over all these years. Thank you for the sacrifices you have made in order for me to pursue a Master's degree.

Thank you to my family for their unconditional love. Despite being far, they always find a way to make me feel close.

Thank you to my dissertation supervisor Prof. Alexandre Bernardino and Eng. Tiago Ferreira from *BladeInsight* for your insight, support and sharing of knowledge that has made this Thesis possible.

Last but definitely not least, to my grandmother who passed away while I was writing this Thesis. You have always been my academical inspiration ever since I was a kid. I am sure you would be proud to witness my accomplishment. I miss you, grandma.

To each and every one of you – Thank you.

Abstract

The world's ever-growing energy needs have directed focus on the deployment of alternative renewable sources, namely wind energy. The blade is one of the most important components for the Wind Turbine to capture energy effectively. Therefore, they need to be continuously monitored. Researchers have introduced a safe, automated, and reliable alternative to inspect the blade: visual inspection through Unmanned Aerial Vehicles, more commonly called drones. In order to detect small damages on the blade, the images acquired need to have high resolution. This Thesis focuses on developing an end-to-end system that automatically produces a high-resolution mosaic of a Wind Turbine blade. The system takes as input the multitude of images acquired by the drone during the inspection and merges them into a single one. However, two major challenges had to be overcome: the distracting background of wind farms and the featureless surface of a blade. To overcome these problems, it is proposed the Graph-Cut algorithm and Contrast Limited Adaptive Histogram Equalization method for Image Enhancement, respectively. The final mosaic obtained by the tailored made solution looks seamless in geometry, i.e. the edges of the blade are aligned and there are no discontinuities. A commercially available software called AutoStitch was not able to produce the mosaic.

Keywords

Image Stitching; Mosaic; Image Segmentation; Image Enhancement; Wind Turbine; GraphCut; Contrast Limited Adaptive Histogram Equalization.

Resumo

As crescentes necessidades energéticas no mundo, direcionaram o foco para as energias renováveis, entre elas a energia eólica. A pá é um dos elementos mais importantes para que a turbina eólica capture energia de forma eficaz. Portanto, precisa de ser continuamente monitorizada. Investigadores introduziram uma forma segura, automática e confiável de inspecionar a pá: inspeção visual através do uso de Veículos Aéreos Não Tripulados, mais comumente chamados de drones. Para detetar pequenos defeitos na pá, as imagens adquiridas precisam de ter alta resolução. Esta dissertação concentra-se no desenvolvimento de um sistema que produz automaticamente um mosaico de alta resolução da pá de uma turbina eólica. O sistema tem como entrada as imagens adquiridas pelo drone durante a inspeção e junta-as numa única. No entanto, dois grandes desafios tiveram de ser ultrapassados: o plano de fundo das imagens extremamente rico em textura e a falta de textura na superfície da pá. Para ultrapassar estes problemas, é proposto que se use o método *GraphCut* para a segmentação da pá e o método *Contrast Limited Adaptive Histogram Equalization* para o aumento de textura na superfície da pá. No mosaico final obtido pela solução desenvolvida observa-se que as linhas das extremidades da pá não têm descontinuidades como era pretendido. O software comercialmente disponível chamado AutoStitch não foi capaz de produzir o mosaico.

Palavras Chave

Mosaico; Segmentação da Imagem; Melhoramento da Imagem; Turbina Eólica; GraphCut; Contrast Limited Adaptive Histogram Equalization;

Contents

List of Figures

List of Tables

List of Algorithms

Listings

Acronyms

AE	Acoustic Emissions
AKAZE	Accelerated-KAZE
AoV	Angle of View
BRISK	Binary Robust Invariant Scalable Keypoints
BRIEF	Binary Robust Independent Elementary Features
CC	Cross-Correlation
CDF	Cumulative Distribution Function
CLAHE	Contrast Limited Adaptive Histogram Equalization
CV	Computer Vision
CNN	Convolutional Neural Network
DIC	Digital Image Correlation
DL	Deep Learning
DT	Destructive Testing
DoF	Degrees of Freedom
DoG	Difference-of-Gaussian
FAST	Features from Accelerated Segment Test
FED	Fast Explicit Diffusion
FoV	Field-of-View
FREAK	Fast Retina Keypoint

GPS	Global Positioning System
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging
LMedS	Least-Median of Squares
LoG	Laplacian-of-Gaussian
MI	Mutual Information
ML	Machine Learning
NDT	Non-Destructive Testing
ORB	Oriented FAST and rotated BRIEF
PCA	Principal Component Analysis
RANSAC	Random Sample Consensus
SHM	Structural Health Monitoring
SIFT	Scale-Invariant Feature Transform
SURF	Speeded-Up Robust Features
SVM	Support Vector Machine
TSURF	Template Speeded-Up Robust Features
UAV	Unmanned Aerial Vehicle
WT	Wind Turbine

1

Introduction

Contents

1.1 Motivation

The depletion of fossil-fuel reserves, stricter environmental regulations and the world's ever-growing energy needs have directed focus on the deployment of alternative renewable sources. Among the various renewable energy alternatives, wind energy is one of the most promising and fastest growing installed alternative-energy production technologies [13, 14]. In fact, in 2018 Europe set the goal that by 2030 at least 32% of their energy needs will be met by renewable energy [15]. In order to achieve this goal, the association WindEurope [1] expects Europe to build 105GW of new wind capacity by 2025 as shown in figure 1.1. This desired expansion bears challenges such as wind turbine monitoring for maintain a profitable and competitive energy cost.

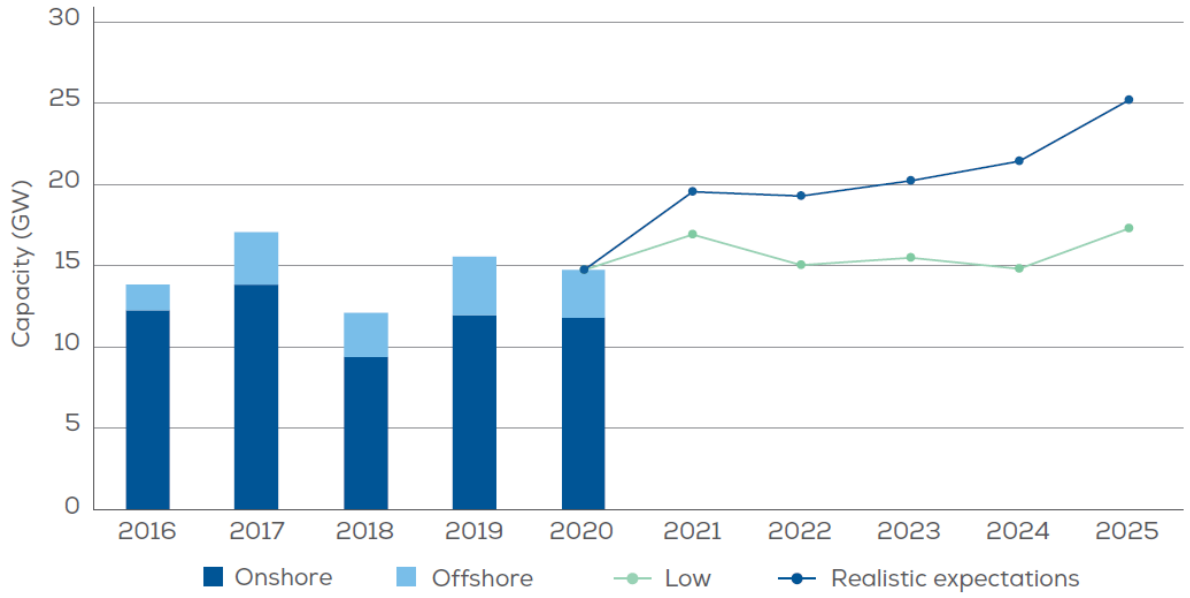


Figure 1.1: New installations in Europe - WindEurope's scenarios [1].

The blade is one of the most important components for the Wind Turbine (WT) to capture wind energy effectively. It is continuously subjected to a combined effect of material aging, structural degradation, and environmental loads such as wind, solar radiation, acid rain, and others [16]. In practice, the erosion at the leading edge reduces the airfoil lift by up to 23% and drag by 100%, reducing its aerodynamic efficiency, having the effect of reducing annual energy production by up to 4% [17]. Therefore, WTs need to be continuously monitored. If damages can be detected and failures predicted earlier, the repair costs and downtime due to maintenance can be driven down and optimized [18].

Historically, traditional maintenance methods involve human inspection [19]. Specialized rope access teams inspect one blade at a time while it is in the down position. This is an expensive and time-consuming process that introduces subjectivity into the results, besides being a life-threatening job.

Alternatively, inspections can be carried out through ground cameras, but the quality and type of data they can obtain is poor, which means damage can go unnoticed. From the danger incurred during manual inspection, to the inability to get full coverage of the turbine surface using ground based cameras, it is evident that both methods have downsides [20].

To overcome these limitations, significant effort and funding has been channeled into finding safe, automated, and reliable alternatives. Hence, researchers have introduced remote inspection solutions employing Unmanned Aerial Vehicles (UAVs). This solution provides a number of benefits. It is safe, as no human personnel are required to climb the turbine. Also, as the UAV is able to freely navigate around the turbine, it is possible to obtain a complete view of the surface. Another significant benefit is the ability to perform consistent and repeatable inspections [20].

Nevertheless, the amount of data acquired in this process raised a new challenge. The manual visual inspection of the resulting plethora of frames is a time-consuming process. Further advancements over the last decade in camera technology and image-processing algorithms have made Computer Vision (CV) a viable option for streamlining the blade monitoring process [21].

1.2 Research Context

BladeInsight is a Portuguese company that has been developing technology for WT blade inspections since 2015. They developed a reliable and repeatable data collection system with autonomous drones, seamlessly integrated with a multi-source analytics platform to best visualize data, extract insights and manage maintenance priorities. Therefore, their business comprises various stages, starting with completely automated drone data collection, and ending with the delivery of insight and maintenance prioritization to the customer.

This Thesis arose from the fact that the technology developed by this company is missing a step in their automatized process. Having acquired the data, Deep Learning (DL) algorithms are employed to detect damages on the blade. However, the frames do not provide any spatial information about the defects. Therefore, afterwards, it requires human intervention for performing a time-consuming task of merging together the thousands of pictures collected from the wind-farm to create a *mosaic*. This merging is necessary to generate a global defect map to assess and locate the defects within the blade. The mosaic generated streamlines the data management, and, ultimately, improves the costumer's visual experience.

1.3 Objectives & Contributions

The main goal of the project is to investigate, develop and implement an end-to-end system to automatically produce a high-resolution image, also called a *mosaic*, of a wind turbine blade. For this purpose,

the system developed should take as input the multitude of images acquired by the drone during the inspection and merge them into a single one. The process should be fully automated, and not require any manual operations. Whereas, the solution should look as seamless as possible in geometry, i.e. the lines along the edges of the blade must be continuous. Accomplishing this objective, it is possible to generate a global defect map for each side of each blade. Such a map would facilitate the locating the damages on the blade, and help in the visual assessment of the global defect pattern.

1.4 Document Outline

The remainder of this report is structured around 5 chapters. Chapter 2 provides a brief historical evolution of the Structural Health Monitoring (SHM) techniques of WT blades. This is followed by an comprehensive theoretical background, and related research on image stitching. The study regarding image stitching applied directly on WTs pointed out the need to study image enhancement techniques and image segmentation methods. Therefore, these two topics are also addressed in this Chapter. Chapter 3 presents details on key components and characteristics of a WT, an explanation on how the dataset was acquired, a description of the dataset itself, and the available metadata. In Chapter 4, the several trials performed before achieving the final implementation are described. Furthermore, the final implementation details of the system developed are introduced. Afterwards, in Chapter 5 a qualitative and quantitative evaluation of the experimental results is presented, along with a comparison with an available commercial image stitching software. Finally, Chapter 6, presents concluding remarks, a reflection on the system limitations, as well as final thoughts on the opportunities for future research.

2

Literature Review

Contents

A comprehensive overview on related previous and ongoing research on the several areas covered by this project frames the presentation of this chapter. First, the chapter gives some context on how SHM techniques on wind turbines evolved over the years - Section 2.1. The most traditional approaches are introduced first, mentioning the existing sensor-based Destructive Testing (DT) methods. Afterwards, the novel Non-Destructive Testing (NDT) techniques, that leverage from the computer vision and deep learning advancements, are presented.

The findings of this literature review pointed out a gap that could receive further attention for generating a global defect map: image stitching on the blade surface - Section 2.2. Thus, a deeper insight on image stitching methods having a general target is presented, highlighting the differences between direct methods - Section 2.2.2 - and feature-based methods - Section 2.2.3. In other hand, when the target of image stitching is specifically a WT blade - Section 2.2.4 -, the Literature drew some attention to the problem of lack of features on the blade surface, and to the importance of removing the background. Therefore, Section 2.3 focuses on two Image Enhancement techniques: Histogram Equalization and Contrast Limited Adaptive Histogram Equalization (CLAHE). Finally, Section 2.4 revises Image Segmentation methods. Figure 2.1 graphically summarizes the scope of the literature review.

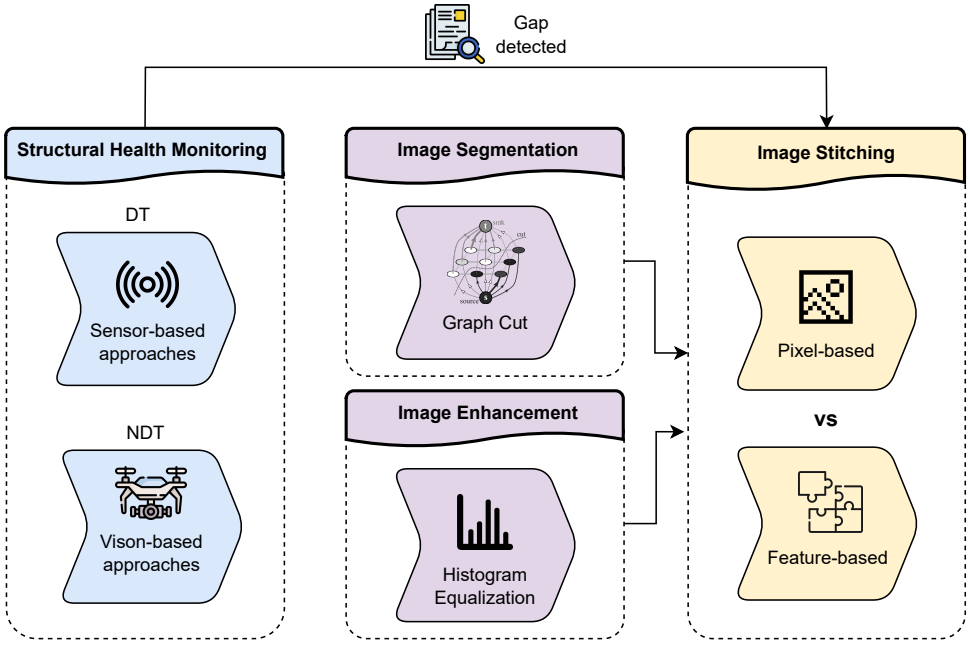


Figure 2.1: Conceptual diagram of the topics covered on the literature review.

2.1 Structural Health Monitoring and existing issues

Consistent SHM allows timely detection of damages on wind turbine blades, which is imperative for minimizing downtime and avoiding catastrophic structural failures [22]. Two basic types of SHM have

been identified in the literature: sensor-based DT and NDT. In this chapter the focus is on the NDT methods because of its recognized benefits. These benefits uncovered a gap related to the need to locate the cracks on the blades. Therefore, multiple images acquired must be stitched, reconstructing a mosaic of the whole blade.

Earlier studies considered monitoring the condition of WT blades by analyzing signals from installed sensors. There are several previous academic researches focusing on the conventional and newly developed sensors for damage detection approaches such as Acoustic Emissions (AE) sensors [23], strain sensors, ultrassonic sensors, and vibration sensors [24]. Ying et al. [25] reviewed the SHM methods for WT blades. The primarily disadvantage of vibration methods is that they require extra sensors and actuator patches to be bonded to the surface of the blade. Automatically, extra load is applied. Therefore, Non-Destructive Testing (NDT) methods are preferable as they do not place further strain on the blade [22].

Despite the different approaches identified, most of the literature reviewed agreed on one idea. Regardless of the sensors employed, details of defects on WT blades still require further visual inspections [24]. UAV inspections were introduced to fill this gap. In an effort to reduce human intervention in processing the extensive amount of data acquired by the UAV, vision-based techniques have received more attention. Recently, computer vision breakthroughs with Machine Learning (ML) have developed and been refined over time. For further information, A. Joshuva et al. and D. Willis et al. [26,27] reviewed the state-of-the-art in wind energy research.

Nonetheless, defect detection in images is a challenging task due to the small size of the cracks, and the existence of noisy patterns on the surface of the blade. To tackle this challenge, Y. Chian and J. Tian [28] propose an approach that breaks a large image that contains multiple separate defects into smaller patches to feed the Convolutional Neural Networks (CNNs). As an alternative, L. Wang and Z. Zhang [24] use Haar-like features to feed stage classifiers selected from a set of base models: the LogiBoost, Decision Tree, and Support Vector Machine (SVM).

Additionally, most of the acquired data contain frames with no defects. This generates unbalanced databases for feeding the classifiers. Researchers were able to overcame the issue of the scarce availability of damage samples through data augmentation (i.e., rotation and flipping) [29, 30].

Another challenge on image processing is the blurred motion of aerial images. As avoiding such low-resolution images for training can affect accuracy, D.Sarkar and S. K. Gunturi [31] use a super-resolution CNN. Image enhancement techniques are also mentioned by J. Zhang et al. [30] as a reliable method for increasing accuracy. In the same article, the authors state that YOLOv3, YOLOv4 and Mask R-CNN are state-of-the-art deep learning algorithms for detecting and classifying defects. Thus, they proposed a new performance evaluation measure and concluded that Mask R-CNN outperformed the others.

After screening the aforementioned literature, a space for improvement has been identified. In none

of the cases, the authors consider an image stitching method as a final step for post-processing the dataset. A standard dataset comprises a vast amount of pictures. The classifier analyses them separately, and is able to detect damage or defects with high accuracy. At the end, despite knowing that the defect exists, it is difficult to pinpoint its location due to the limited Field-of-View (FoV) of each frame. In case the frames are merged together into a mosaic that covers the whole blade, it is easier to locate the defect within the investigated area. Besides helping to locate the defect, such approach could streamline the data management, and provide a better visual experience to the costumer. Therefore, in the following section, a theoretical background regarding panoramic image stitching is introduced, as well as several representative studies.

2.2 Image Stitching and its value in SHM

Several authors consider that image stitching is the same as image mosaicing, whereas others consider that stitching is a step in the pipeline of image mosaicing. Regardless the name, there is consensus that this problem can be defined as the process of merging multiple images of the same scene with overlapping FoVs to produce a high-resolution image [32]. A number of image mosaicing reviews and surveys have been analyzed in the literature [5, 23, 33–35].

Even though in recent years the state-of-the-art indicates advancements in this research area, image mosaicing still remains a challenge. One reason is there is not a common method for stitching all kinds of images due to their inherent complexity [23]. In addition, despite the plethora of existing methods for panoramic image stitching, there is still no public dataset for such purpose [36], which makes it harder to benchmark them. Furthermore, mosaicing in the presence of images with significant parallax, varying illuminance and exposure conditions is still a challenge [33].

Nowadays, this technology is widely available as it is embedded in almost every smartphone on the market - the panoramic camera feature [37–39]. However, before smartphones had the processing capability for accomplishing it, researchers already used image mosaicing techniques for processing aerial imagery [40]. When the first satellites started sending Earth images back, mosaicing became a need. Later, the arising of UAVs in the research area, provided a massive breakthrough. UAVs allowed surveillance [41], environmental monitoring [42], and disaster assessment [43]. More recently, image mosaicing technology has become a research hot spot in the domain of medical image processing for diagnosis [44]. An application in which mosaics are specifically useful is in the diagnosis and treatment of retinal diseases [35]. Nevertheless, the wide FoV provided by panoramic images is demanded in another number of applications such as remote sensing [45], object tracking [46], and large infrastructures monitoring, which is the scope of this report.

2.2.1 Image Stitching Overview

This section analyses the various stages that image mosaicing comprises in order to implement it later on. For such purpose, the book written by D. Capel [12] has been studied. There are three important factors to consider when constructing a mosaic: the estimation of a set of geometric transformations that align the images; the choice of reprojection manifold on which the images are composited; and, finally, the choice of algorithm for blending the overlapping images. Therefore, the problem of image mosaicing can be broken down into three steps: registration, reprojection, and blending [33, 38]. The flow of this process is depicted in Figure 2.2, where the output from one sub-process becomes the input to the next.

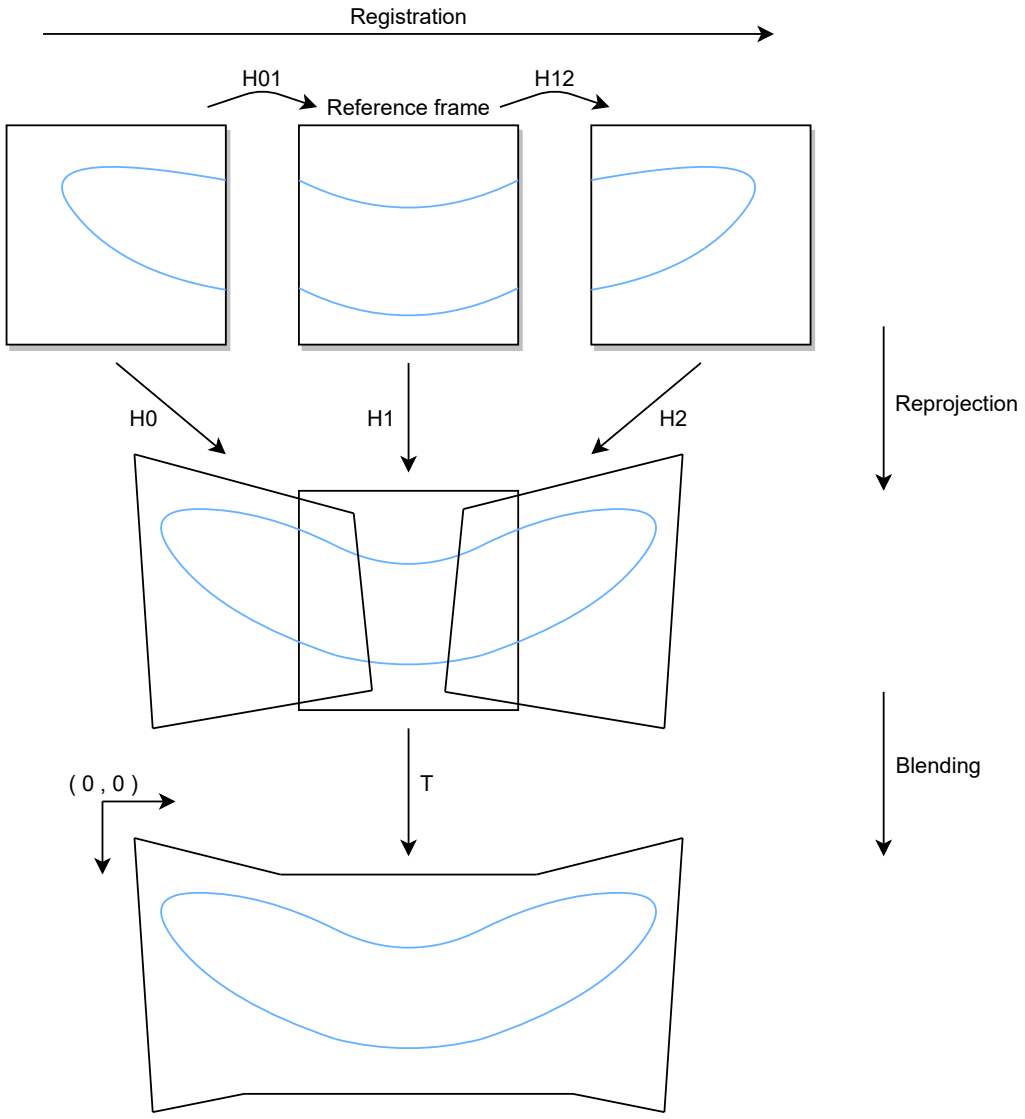


Figure 2.2: Flow diagram of a general image mosaicing process, outlining the three basic steps: (1) registration; (2) reprojection; and (3) blending. In this example, the middle frame is selected as the reference, and hence $H1$ is the identity. The rendering transformation T simply shifts the origin. Source: [12].

Some authors include a primary step before the registration - the calibration [32, 47]. The goal of this step is to estimate the extrinsic and intrinsic camera parameters. Afterwards, image registration is performed to place the images under the same coordinate system. This core step is accomplished by computing the geometric transformations, which align the images, with respect to a reference image within the set [33]. Reprojection follows, which is based upon the registration. The transformations computed in the previous step are employed, projecting every point from each frame onto the global coordinate system. Finally, blending algorithms are applied to visually minimize the seam in the boundaries between images. Thus, it makes the transition between frames look more natural. In the following sections, these three basic steps will be further analyzed.

2.2.1.A Image Registration

It is essential in the process of mosaicing to accurately align the dataset, i.e. to estimate the point-to-point correspondence between images in the input sequence. This is known as registering the images. The problem can be stated as follows: given two different views of the same scene, for each point in one view find the corresponding point in the second view, which is the same actual point in the scene. At the end, the objective of image registration is to register every image into a global coordinate frame which contains the whole scene [12]. Usually, one of the N input images is selected to be the reference and the whole set is projected to be axis-aligned with it. Thus, the transformation between the global frame and the reference image is simply the identity - the transformation $H1$ in Figure 2.2 serves as an example.

Methods for obtaining registration fall into two broad categories: direct methods that compute a transformation which optimizes some measure of photometric consistency over the whole image; and feature-based methods that use a sparse set of corresponding image features (e.g. edges and corners) to estimate the image-to-image mapping. In the following sections are further details on this two categories - 2.2.2 and 2.2.3, respectively.

2.2.1.B Image Reprojection

After registration, every point in every image can be transformed to a point in the global frame. The set of all images and geometric transformations comprises the mosaic representation of the scene. In order to actually render an image from the mosaic representation it is necessary to supply a further transformation which maps points in the global frame to points in the rendered image. In some instances, this mapping may be as simple as a similarity transform (scaling and translation), or may be a more complicated transformation, such as a synthetic camera rotation or cylindrical polar mapping. Having chosen a rendering transformation, each image is then warped into the frame of the rendered image [12].

2.2.1.C Blending

The final stage in rendering from the mosaic is to blend the images together in their overlapping parts. If the significant differences between images are not corrected, this can give rise to unsightly seams in rendered mosaics. Common methods of image blending include simple averaging of intensity values, feathering and temporal median filtering. Nevertheless, better results may be obtained if photometric registration is performed prior to rendering the mosaic [12]. For the purposes of this project, image blending will be disregarded, as the final goal is to visually assess the defects on the WT and not to obtain a seamless stitching.

2.2.2 Pixel-based Stitching

Pixel-based stitching methods - often called *area-based* [5] or *direct* [34] methods - compare all the pixel intensities of the images with each other, directly minimizing pixel-to-pixel dissimilarities [32]. The first step to employ a direct method is to choose a suitable error metric to compare the images. Once this has been established, an appropriate search technique must be devised. The most straightforward way is to exhaustively attempt all possible alignments. However, this bears a major drawback: it is too slow. Therefore, researchers developed hierarchical coarse-to-fine techniques based on image pyramids. Alternatively, Fourier transforms can be used to speed up the computation. To get sub-pixel precision in the alignment, incremental methods based on a Taylor series expansion of the image function are often used. These can also be applied to parametric motion models [34].

2.2.2.A Traditional methods

Zitova and Flusser [5] have classified area-based methods into three subcategories: Cross-Correlation (CC), Fourier and Mutual Information (MI). These methods are described in Table 2.1, along with their main advantages and disadvantages.

2.2.2.B Geo-referenced stitching

Image-based approaches only exploit image information to compute the transformations. Alternatively, a very simple and naive approach is to align the images based on the camera's position. Direct image stitching can be implemented through geo-referenced images that combine metadata from multiple sensors: Global Positioning System (GPS), Inertial Measurement Unit (IMU), and Light Detection and Ranging (LiDAR). Such approaches have been widely used for aerial image mosaicing employing UAVs. Generally it can be implemented in two ways: *offline* [51–53] where the mosaicing is accomplished after the data is acquired, or in *real-time* applications [54–56], such as virtual reality, or disaster assessment. In this literature review it was given more relevance to the *offline* methods to serve the purposes of this

Table 2.1: Categories of area-based methods for feature matching identified by Zitova and Flusser [5].

Method	Brief Description	Advantage	Disadvantage
Cross-Correlation [48]	It is a measure of similarity computed for window pairs from the sensed and reference images. The window pairs for which the maximum is achieved are set as corresponding ones.	Easy hardware implementation makes them useful for real-time applications. Suitable for samples where reference and sensed images have very similar intensity functions.	The flatness of the similarity measure maxima (due to the self-similarity of the images) and high computational complexity.
Fourier [49]	Fourier-based alignment relies on the fact that the Fourier transform of a shifted signal has the same magnitude as the original signal but linearly varying phase.	Strong robustness against the correlated and frequency-dependent noise and non-uniform, time-varying illumination disturbances.	Only useful for sets of images with large overlapping fields.
Mutual-Information [50]	Represent the leading technique in multimodal registration. Registration of multimodal images is a difficult task, but often necessary to solve, especially in medical imaging.	Particularly suitable for registration of images from different modalities, or when reference and sensed images have statistically dependent intensity functions.	Performance affected by local intensities variations. Spatial information ignored.

project. Furthermore, recently geo-referenced stitching was also used for Structural Health Monitoring (SHM) purposes, namely for assessing the state of a bridge [57].

Previously, the imagery was acquired by satellites where the altitude is large enough to approximate the surface to a plane. However, the introduction of UAVs has posed some challenges, because when flying at low altitude the assumption of a planar surface is no longer true [58]. Many alignment methods based on minimizing registration errors would result in a significant accumulation of perspective distortions for hundreds of images acquired from wide-range regions [53]. Furthermore, due to their light weight, small-scale UAVs are vulnerable to wind gusts. Despite the high dynamic control systems embedded to achieve a stable flight, a perfect nadir-view of the images cannot be provided.

In order to overcome these challenges, several studies found reliable solutions. For instance, Yahyanejad et al. [58] proposed a hybrid incremental mosaicing approach that combines metadata-based and image-based methods. In fact, if the position provided by the GPS is accurate and the views are completely Nadir, then the hybrid algorithm is reduced to a position-based approach. The imagery is acquired by an UAV, that flies over a target area along a pre-planned route. These images are transmitted to a ground station and are incrementally stitched to produce a *mosaic* of the target area.

Later, Xie et al. [53] and Xia et al. [59] emphasized the importance of finding the optimal reference image which minimizes the total propagated error. In [53] the authors do not exploit pose parameters. To select a reasonable reference image as the projection plane, they first divide all the images into two groups (stable group and unstable group) according to their registration error under the affine model. The reference image is selected from the stable group, and each unstable group image is locally attached to

a stable group image via a homography.

Concerning SHM applications, Saleem et al. [57] proposed an instant and automatic pipeline for damage identification and localization. Their approach uses an image capturing and geo-tagging system, followed by Deep CNNs for crack detection. The damages extracted by the CNN are instantly transformed into a global bridge damage map, with geo-referencing data acquired using the image capturing and geo-tagging system.

Figure 2.3 depicts the difference between the Nadir view more commonly used in the aforementioned approaches, and the Non-Nadir view applied in utility inspection, such as this project and the bridge inspection [57]. Classically, in remote sensing the camera is pointing down at the Earth. In this case, being the Earth the target, and knowing the exact location of the camera (longitude, latitude, altitude), it is easy to map the points into a global coordinate system. As opposed to the case in study, where the camera has a Non-Nadir view, and thus the location of the target is unknown, i.e. the location of the blade within the frame. Nonetheless, it is possible to extrapolate this information, as long as the position of the camera and the distance to the target is provided.

2.2.2.C Challenges

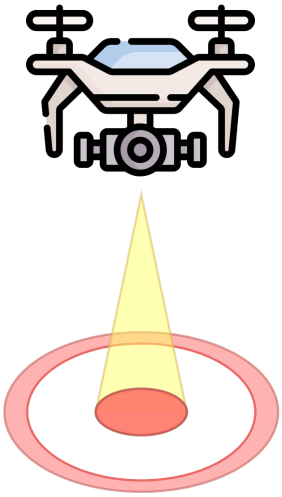
The main advantage of direct methods is that they make optimal use of the information available in image alignment, as they measure the contribution of every pixel in the image. Furthermore, assuming a Gaussian noise model, they properly weight the contribution of different pixels, e.g., by emphasizing the contribution of high-gradient pixels. Nonetheless, these bears some drawbacks, such as the ones found in the literature [5, 32]:

1. **Low overlapping fields:** these algorithms are suitable for samples with large overlapping fields.

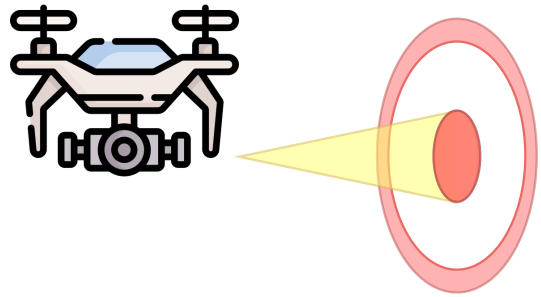
2. **Sensitive to intensity changes:** not suitable for situations when illumination changes are expected or multi-sensor analysis is demanded, as these methods compare pixel-to-pixel intensities. Therefore, reference and sensed images must have 'similar' (CC-like methods) or at least statistically dependent (MI methods) intensity functions.

3. **Can only handle translations:** from the geometric point of view, only shift and small rotation between the images are allowed when using area-based methods (although area-based methods can be generalized to full rotation and scaling, it is practically meaningless because of an extreme computational load).

4. **Convergence on local minima:** direct methods may not converge to the optimal solution in the presence of local minima. For a reliable performance, these methods are dependent on an accurate initialization.



(a) Nadir view.



(b) Non-Nadir view.

(c) Example of image acquired from (a). Source: [60]

(d) Example of image acquired from (b).

Figure 2.3: Comparison of the Nadir view used by classical remote sensing approaches and the Non-Nadir view necessary for utility inspection. On the left side is an example of a Nadir view where the camera pointing down at Earth, and on the right in an example of a Non-Nadir view where the camera is capturing the blade sideways.

5. **Limited range of convergence:** considered the biggest disadvantage. Even though hierarchical (coarse-to-fine) estimation frameworks can be applied, in practice it is hard to use more than two or three levels of a pyramid before important details start to be blurred away.

All in all, area-based methods are preferably applied when the images have not many prominent details and the distinctive information is provided by gray-levels/colors rather than by local shapes and structure. Methods that give further importance to local shapes and structures are introduced next in section 2.2.3.

2.2.3 Feature-based Stitching

In contrast to the area-based methods, the feature-based ones do not work directly with image intensity values. Feature-based approaches aim to determine a relationship between frames through distinct features extracted from the processed images [32]. First, it is important to distinguish between two concepts that will be further explored in the following sections: keypoint and descriptor. A keypoint is an important and distinctive point of interest, such as corners, lines, and edges; whereas, a descriptor is a feature vector containing essential characteristics that represent each keypoint.

2.2.3.A Pipeline

The traditional algorithms of image registration mainly include the fours steps that are illustrated in Figure 2.4: (1) Target Selection, which defines the fixed input that the other input is going to be aligned to; (2) Feature Extraction, which computes the keypoints and extracts local invariant descriptors using using the algorithms introduced in Table 2.2; (3) Feature Matching to find correspondences among the features extracted in the previous step and estimate the respective transformation matrices; and (4) Transformation Model Estimation, which is the process of estimating the transformation model that converts one image into another.

As in this project the input images are already sorted according to the data acquisition, the first step of the pipeline - Target Selection - can be disregarded. Thus, in the following subsections,the remaining steps are explained in detail.

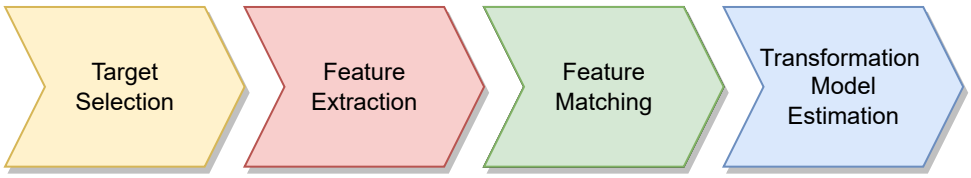


Figure 2.4: Flow diagram of a general image registration process, outlining the four steps: (1) Target Selection; (2) Feature Extraction; (3) Feature Matching; and (4) Transformation Model Estimation.

2.2.3.B Feature extraction

In the literature, many image feature detection and description algorithms have been proposed, among which Scale-Invariant Feature Transform (SIFT) [61], Harris Corner Detector [62], Binary Robust Independent Elementary Features (BRIEF) [63], Fast Retina Keypoint (FREAK) [64], Features from Accelerated Segment Test (FAST) [65], Speeded-Up Robust Features (SURF) [66], KAZE [67], Binary Robust Invariant Scalable Keypoints (BRISK) [68], and Oriented FAST and rotated BRIEF (ORB) [69] are some popular ones. SIFT [61] - the most well-known algorithm and one of the first ones to be introduced - is rotation, and scale invariant, as well as robust against noise. It is highly accurate; however, the trade-off

is also high computational burden. SURF [66] improves the computation time of SIFT [61] by using an integral image for fast local gradient computations on an image. In an effort to create algorithms suitable for real-time applications, that require less computational time, researchers have introduced binary descriptors. For instance, ORB blends the modified FAST detection and direction-normalized BRIEF methods, which are both based on Hamming distances and binary operations. It is an extremely fast operation, while sacrificing very little on performance accuracy. A number of detailed analysis and comparisons of these feature detection and description algorithms can be found in the literature [6–10], where the authors assessed the different algorithms and tried to set a benchmark for researchers. A summary of the gathered information is presented in Table 2.2.

Table 2.2: Analysis and comparison of the most popular keypoint detection, and descriptor methods, including a brief description followed by the main advantages and disadvantages. Inspired in the literature [6–10].

Method	Brief Description	Advantage	Disadvantage
SIFT [61]	Based on the Difference-of-Gaussian (DoG) operator which is an approximation of Laplacian-of-Gaussian (LoG) [70].	Highly accurate. Invariant to rigid translation, scaling, and rotation.	Large computational complexity limits the use in real-time applications.
HARRIS [62]	Detects features through corners and edges. It uses a normalized cross-correlation of intensity values to match them.	Invariance to illumination changes and to image noise.	Needs prior knowledge window size. Only good for moderate changes in scale and rotation. Only detector, not descriptor.
BRIEF [63]	Randomly selects pairs of points around a keypoint to create a binary descriptor. Uses pixel intensity comparisons between the pairs.	Fast construction and matching. High recognition rates.	Not robust to large in-plane rotations.
FREAK [64]	Uses a circular retinal sampling grid with a higher density of points near the center, and compares the pixel intensities.	Coarse-to-fine structure speeds up the matching using a cascade. Scale and viewpoint invariant.	It is only a descriptor, not a detector.
FAST [65]	Template based corner detector scheme, that tests a number of brighter or darker pixels in a ring around a given point.	Computationally efficient, suitable for real-time applications because of this high-speed performance.	Not robust to high levels of noise. It is dependent on a threshold.
SURF [66]	Based on the determinant of Hessian Matrix and it exploits integral images to improve the computational efficiency of feature detection.	Fast computation, good for real-time applications.	Poor performance under certain transformations.
KAZE [67]	Based on non-linear diffusion filtering. The maxima of the detector responses are picked up as keypoints using a moving window.	One of the fastest ones. Low computational cost.	Lower accuracy in feature detection.
BRISK [68]	Detects corners using the AGAST [71] algorithm and filters them with the FAST [65].	Low computational burden. Provides better performance than SURF and comparable accuracy with SIFT.	Outperformed by ORB in keypoint verification.
ORB [69]	Combines features of FAST [65] and BRIEF [63] for feature extraction and description.	Fast computation speed. Efficient memory usage. Highly accurate.	Outperformed by BRISK in feature matching.

Several variations were suggested to the methods introduced in Table 2.2. For instance, SIFT is improved in [72], known as PCA-SIFT. This variation accepts the same input as the standard SIFT descriptor. Principal Component Analysis (PCA) conducts dimensionality reduction. Therefore, PCA-SIFT consumes less time than SIFT with the cost of lower robustness to scaling [23]. RootSIFT [73]

differentiates from SIFT by using a Hellinger distance instead of the standard Euclidean distance to measure the similarity between its descriptors. Another example is Accelerated-KAZE (AKAZE) [74], which is based on non-linear diffusion filtering like KAZE [67]. However, its non-linear scale spaces are constructed using a computationally efficient framework called Fast Explicit Diffusion (FED). Finally, Template Speeded-Up Robust Features (TSURF) - an improvement of SURF - is proposed in [75]. This approach has proven to improve the stitching speed, while maintaining the precision [23].

Overall, SIFT and BRISK are found to be the most accurate algorithms, while ORB and BRISK are most efficient [6]. Nevertheless, the choice of the feature detector depends on the problem, as there is not one single method that suits all situations [32]. For instance, the most accurate algorithms are not proper for real-time applications due to their computational time; whereas for *offline* processing, accuracy is often preferred over speed. Another important consideration is whether there is a leverage or not on striving for orientation and scale invariance. In some cases, neglecting orientation estimation can lead to a better performance both on computational load and accuracy.

2.2.3.C Feature Matching

The aforementioned methods extract the feature descriptors from images. At this intermediate stage of the pipeline, the goal is to establish correspondences between the key points. In order to do so, a descriptor from one image needs to be matched with a descriptor from another image to find the closest matching feature. This task can be done in various ways, but the most accepted way is to use the Euclidean distance between these descriptors. The following equation computes the Euclidean distance between two vectors:

$$|A - B| = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \tag{2.1}$$

where A is a descriptor of Image 1, and B a descriptor of Image 2. Likewise, a_i is an element of the descriptor A and b_i an element of B. In python the *numpy* library provides this functionality using linalg.norm [76].

Outlier Rejection - Random Sample Consensus

Thus far, the interest point features have been extracted and the respective putative correspondences have been computed. Next is the elimination of the resulting outliers in the initial correspondence set. For such purpose, Random Sample Consensus (RANSAC) [77] is the most commonly used method, as it is extremely simple but highly effective. This is an iterative three-step procedure, that can be repeated N times. Algorithm 2.1 describes those steps. A question that needs to be asked, however, is how many times should this algorithm be iterated. The number of iterations, N, required to succeed with

probability, p, having an outlier ratio, e, and a sample of points of size s is given by:

$$N = \frac{\log(1-p)}{\log\left(1-(1-e)^s\right)} \tag{2.2}$$

where, for instance, when computing an Homography the value of s is 4, as it is the minimum number of points necessary to compute an Homography.

Algorithm 2.1: The main steps in RANSAC algorithm to estimate the inliers in a set of correspondences between two images. Inspired in: [12]

Result: inlier set.

for N **do**

1. Select a random sample of four correspondences and compute the homography H.
2. Calculate a geometric image distance error for each putative correspondence.
3. Compute the number of inliers consistent with H by the number of correspondences for which the distance error is less than a threshold.

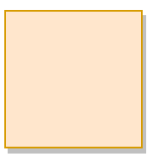
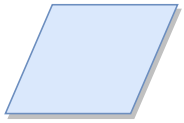
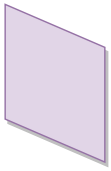
end

Choose the set with the largest number of inliers.

2.2.3.D Transformation Model Estimation

After computing the correspondences between frames, and removing the outliers, everything is set for estimating the transformation model. This model will be able to project the corresponding points of one image into another. First, it is important to acknowledge the basic image operations available in Table 2.3.

Table 2.3: Image basic operations, the respective transformation matrices, and an example.

Identity	Scaling	Translation	Rotation	Shear (x)	Shear (y)
$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & \lambda_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ \lambda_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
					

The basic operations are further analyzed having in mind the purpose of this project: stitching a WT blade. The **identity** operation is always useful for placing the first image, also known as reference. The

scaling operation helps when the distance to the blade varies. The **translation** operation is required for every frame, as the drone acquires the frames moving parallel to the blade. In other hand, it is most likely that the **rotation** and both **shear** operations will not be necessary, as the drone avoids rotations. These operations presented in Table 2.3 can operate alone, or they can be combined into the transformations presented in Table 2.4. For instance, the **Euclidean** transformation is the combination of a rotation and a translation operation. Furthermore, the transformations presented in Table 2.4 are sorted according to their range. i.e., an Euclidean transformation is a type of Affine transformation, while an Affine transformation is a type of Homography. Therefore, the Euclidean transformation as an Affine transformation, preserves all Affine properties, such as incidence and parallelism. In addition, preserves lengths of line segments and angles between two line segments. Because of this, a Euclidean transformation is also called a rigid motion. Given its properties, the Euclidean Transformation is not suitable, as it does not provide a rescale option. The **Scaled Euclidean** transformation tackles this problem and provides a scaling factor s. In other hand, the key differences between an **Homography** and an **Affine** transformation are:

- Homography: lines are mapped to lines, but parallelism may not be kept;
- Affine: collinearity and parallelism are both kept.

Table 2.4: Transformations, respective matrices, number of DoF, and their properties.

Name of the Transformation	Transformation Matrix	Number of DoF	Properties		
			Size	Parallel Lines	Angles
Euclidean	$\begin{bmatrix} \cos\theta & -\sin\theta & tx \\ \sin\theta & \cos\theta & ty \\ 0 & 0 & 1 \end{bmatrix}$	3	X	X	X
Scaled Euclidean	$\begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot Euclidean$	4		X	X
Affine	$\begin{bmatrix} a_{11} & a_{12} & tx \\ a_{21} & a_{22} & ty \\ 0 & 0 & 1 \end{bmatrix}$	6		X	
Homography	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}$	8			

In addition, for computing an Homography it is required 4 points since each point has x and y coordinates and there are total 8 DoF to solve H for a unique solution. Similarly, for the Affine transformation it is required 3 points, and for the Scaled Euclidean only 2. All things considered, the Affine Transformation might be an overkill for the purpose of the project, since the drone moves parallel to the target, no

rotation is required and the projection is planar. Whereas, the Euclidean Transformation is not enough. Thus, the best approach suggested is the **Scaled Euclidean** Transformation.

2.2.3.E Challenges

Traditional image stitching methods based on hand-crafted features are effective for constructing a panorama from a sequence of images in the case when there are sufficient reliable features in the scene. However, these methods are unable to handle low-texture environments where no reliable feature correspondence can be established. Furthermore, the accuracy of most methods is dependent on the quality of the dataset, that should fulfill some requirements such as the ones mentioned in [78]:

1. **Overlapping fields:** enough overlapping areas should be maintained between the input image sequences.

2. **Illumination variances:** no severe exposure, intensity, color differences between the input images.

3. **Resolution:** the resolution of stereo input image pairs should be restricted to a certain limit (generally less than $2K$) so that the image-stitching operation can be efficiently performed.

4. **Geometrical integrity:** the orientations and resolutions of every image in the input image sequences should be similar, i.e it is important to maintain geometrical integrity among the input images.

Nonetheless, the different challenges of traditional registration approaches, have lead into one general limitation: the lack of generalization of these algorithms [79]. Depending on the problem, different methods should be addressed [32].

2.2.4 Image Stitching on a Wind Turbine blade

The number of papers in the literature tackling image stitching on WT blades is sparse. Li et al. [80] proposed a feature-based image stitching method for blades of WT, along with background removal preprocessing. The authors realized that the background and blade parts in images of WTs are monotonous. Therefore, it is possible that feature points in these two parts are mismatched. They conducted experiments that compared their approach with a preprocessing blurring method, and concluded that their method was more robust. A question that needs to be asked, however, is whether their method would work on a different blade. The blade in their project had red lists, which is where most features were extracted for the matching stage. Their study would have been more useful if the authors had conducted experiments on the most typical blades that are completely white.

Another alternative found on the literature is Digital Image Correlation (DIC) [81, 82]. This approach has proven to be useful to assess the state of deformation and strain of a blade. Nonetheless, one of the limitations of this method is the need to apply a trackable pattern on the surface of the blade, to overcome the lack of features issue. This implies the cost and time demanding task of taking down every blade in order to apply the pattern, which is not feasible for the project in hands. An alternative for tackling the lack of features is suggested in the following Section.

2.3 Image enhancement techniques

In the scenario of low texture environments, it is useful to study image enhancement techniques. These techniques cover ways of manipulating the image so that the result is more suitable than the original image. They can modify the spacial domain, the frequency domain, or a combination of both. This section will cover spatial domain techniques, which involve direct manipulation of pixels.

Histograms are a powerful technique used to better understand image content. This section is based on the book [83], where F. Alberto provides a theoretical explanation on Histograms and how to leverage from the *OpenCV* library. An image histogram reflects the tonal distribution of the image, plotting the number of pixels for each tonal value. Therefore, a grayscale image with pixels in the range of $[0, K-1]$ will produce an histogram with exactly K entries. These entries are usually called *bins* in histogram terminology. *OpenCV* uses *histSize* to refer to *bins*. For example, for 8-bit grayscale images, $[K = 256]$ $(2^8 = 256)$, the intensity values are in the range $[0, 255]$ and each entry on the histogram is defined as follows:

$$h(i) = p_i(i \in [0, 255]) \tag{2.3}$$

Being p_i the number of pixels with intensity i.

OpenCV provides two alternatives for feature enhancement, using histograms: Histogram Equalization and CLAHE. These methods are explained in detail in the following.

2.3.1 Histogram Equalization

The *OpenCV* library provides the function *cv2.equalizeHist()* which performs Histogram Equalization in *python*. This function normalizes the brightness as well as increases the contrast of the image. Equalization involves the mapping of one distribution into another in order to spread the pixel intensity values over the whole range. For such purpose, the remapping is accomplished by the Cumulative Distribution Function (CDF). For the histogram $H(i)$, its cumulative distribution $H'(i)$ is:

$$H'(i) = \sum_{0 \leq j < i} H(j) \tag{2.4}$$

To use this as a remapping function, we have to normalize $H'(i)$ such that the maximum value is 255 (the maximum pixel intensity value found on the image). Therefore, the remapping procedure to obtain the intensity values of the equalized image is given by:

$$equalized(x, y) = H'(src(x, y)) \tag{2.5}$$

2.3.2 Contrast Limited Adaptive Histogram Equalization (CLAHE)

Overenhancement results in the appearance of darker regions in the enhanced image. From this limitation, arises the concept of CLAHE. CLAHE is a variant of adaptive histogram equalization in which contrast amplification is limited. This algorithm operates on small regions in the image, called *tiles*, rather than the entire image, creating several histograms. The histograms of neighboring *tiles* are then combined using bilinear interpolation to remove the artificial boundaries [84].

When applying CLAHE there are two parameters to tune:

- *clipLimit* - sets the threshold for contrast limiting. By default is 40;
- *tileGridSize* - sets the number of tiles in the row and column. By default is 8x8.

2.4 Image Segmentation

Based on the current literature research, it was found, that image stitching requires a pre-process method, so called image segment. In the following the general aim, an overview of existing methods, and the required steps are given.

Image segmentation is the process of partitioning an image into multiple segments. It can be described as a filter that extracts which information is relevant in a picture and discards what is not. Applied on the given problem the goal is to split the foreground (blade) and the background (surrounding environment). Therefore, only binary methods will be discussed.

In the last 20 years the CV community has developed a broad range of algorithms for image segmentation. There exist multiple methods for partitioning an image into two segments: “object” and “background”. Some examples are the following: graph cut [2], grab cut [3], snakes [85], active contours, geodesic active contours [86], “shortest path” method, etc.

Boykov and Jolly [2] developed graph-cut, an interactive method that borrows tools from graph theory to partition images into foreground and background. The process is summarized in Figure 2.5 and comprises three stages: (1) first the user selects several pixels from both the object and the background, also called seeds; (2) then the graph is constructed based on image information; (3) once the graph is constructed, the goal is to cut it with minimum cost.

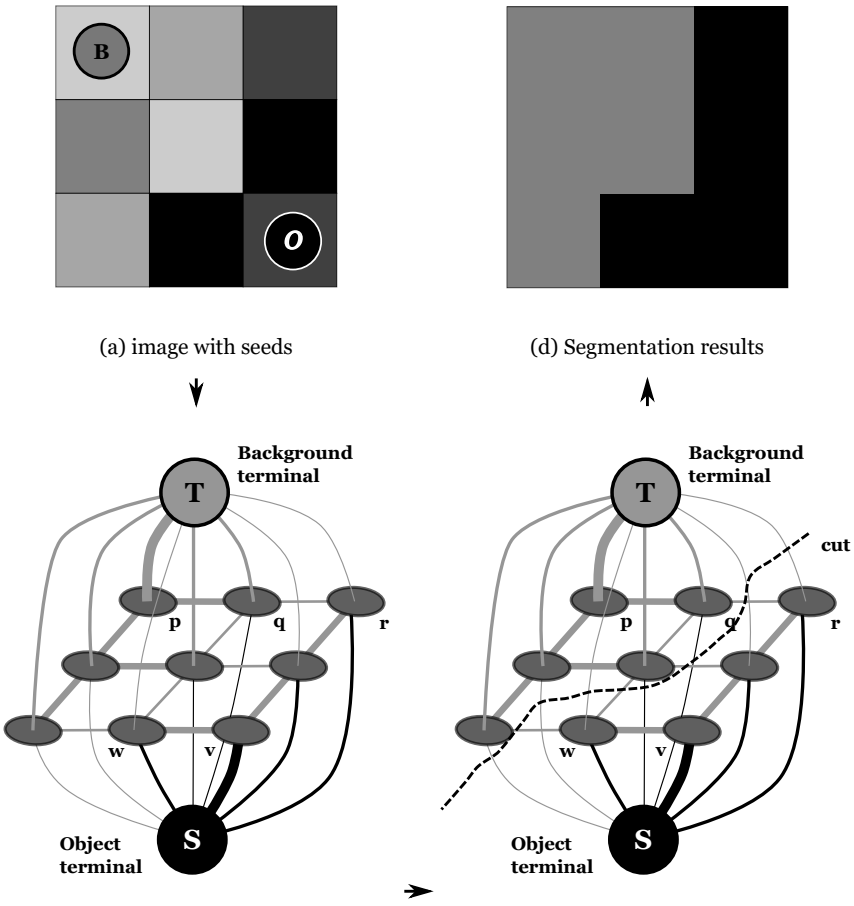
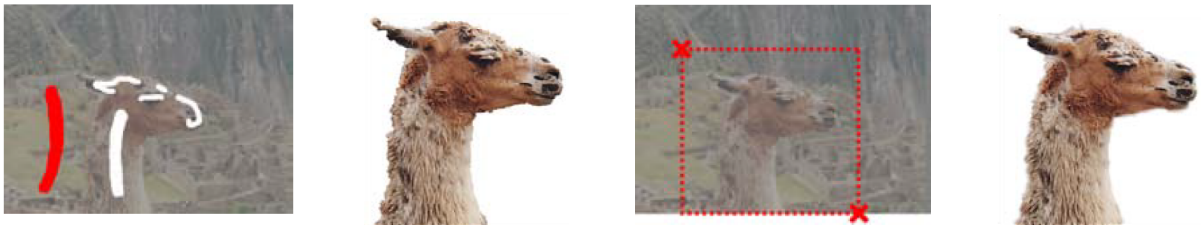


Figure 2.5: Illustration of a simple 2D example of graph-cut for a 3 x 3 image. The seeds are $\mathcal{B} = \{p\}$ and $\mathcal{O} = \{v\}$. The thickness of each edge reflects the cost. Less expensive edges are attractive choices for the minimum cost cut. Source: [2].

Every pixel corresponds to a node of the graph. In addiction, there are two terminal nodes: the object (source - S) and the background (sink - T). Every node is connected to both terminal nodes with a measurement of the probability of belonging to the foreground or background - t links. Besides, the nodes are connected among themselves - n links. A question that arises, however, is what is the weight on those edges. The weight should promote pixels that are similar to stay together on the same segment. There are several ways to accomplish this, for instance, weights can be proportional to the gradient, or to the difference of the pixel values in the neighborhood. Once the graph is constructed, maxflow is invoked to cut the graph with minimum cost.

The main advantage of the method graph cut, compared to the other aforementioned methods, is that it provides a globally optimal solution for an N-dimensional segmentation [2]. Other common methods either offer no clear cost function, or only an approximated solution, which provides a local minimum. This local minimum can be far from the global solution.

Rother et. al [3] developed an algorithm with regional cues based on Gaussian mixture models for improved interactivity, so called grab-cut. It extents the graph cut method in three main aspects. Firstly, an interactive optimization is enabled allowing the user to add further seeds to the initial ones, when the result is not the expected. The process continues without running the whole computations again. Secondly, grab cut simplifies the user interaction for giving the seed. Since, it is only necessary to mark a square for the desired object. This is presented in Figure 2.6. Thirdly, a robust algorithm to estimate the foreground within the selected square was developed.



(a) Graph-cut: user input. **(b)** Graph-cut: result. **(c)** Grab-cut: user input. **(d)** Grab-cut: result.

Figure 2.6: Comparison of the methods graph-cut and grab-cut. Subfigures (a) and (c) show the user input required to complete the segmentation. Subfigures (b) and (c) show the respective results. In (a) the background is represented in red and the object in white, whereas in (c) the user input is only a square surrounding the object. Grab-cut appears to simplify the user input without compromising the results. Source: [3].

2.5 Takeaways

Feature-based methods alone might not appropriated for the challenging task of mosaicing a WT blade. As mentioned in 2.2.3, feature-based methods fail when no feature correspondence can be established due to lack of texture. In other words, feature-based methods are expected to work if only feature correspondences can be found between images. In a WT context, the newer the blade, the cleaner and smoother its surface is, which means that it has virtually no texture. For facing such limitation, image enhancement techniques seen in Section 2.3 seem to be a valid approach.

Analyzing Table 2.4, it is possible to predict which possible transformation best suits the project in hands. The Euclidean Transformation is not suitable, as it has no factor of scale. While acquiring the frames the drone tries to remain at the same distance to the blade but it is still susceptible to variations. The Affine Transformation might be an overkill for the purpose of the project, since the drone moves parallel to the target and the projection is planar. Thus, the best approach suggested is the Scaled Euclidean Transformation.

Besides, as mentioned in section 2.2.4, the majority of the pictures of an in-service WT in wind-farms have a monotonous background. The distracting background and the lack of keypoints on blade may lead to mismatching features. Therefore, background removal techniques for preprocessing the images

should be also considered.

As stated in section 2.2.2, pixel-based methods require an accurate initialization to avoid converging to a local minima and to optimize the computational time. This initialization can leverage from the camera pose measurements referred to in section 2.2.2.B. The company *BladeInsight*, besides the high-resolution frames, also provides IMU, GPS, and LiDAR metadata. Therefore, having knowledge of the camera position and the distance to the target, each image can be directly mapped for panoramic image stitching.

Despite that, it was verified that no one has ever accurately applied image stitching on the surface of a wind turbine blade, and neither tried to generate a global defect map out of it, after the classification algorithms are employed. Hence, the rest of the work to be carried out will try to evaluate the true value and effectiveness of applying image stitching to a blade employing the state-of-the-art techniques. It is also important to keep in mind that the final purpose is to accomplish a better overall pipeline regarding the structural health monitoring of a wind turbine.

3

Data Acquisition

Contents

In this Chapter is conducted a study on the way data is acquired by the company *BladeInsight*. First, in Section 3.1, the target - a Wind Turbine (WT) - is inspected by its characteristics: color, texture, shape, and size. Second, in Section 3.2 the hardware specifications are stated and the setup employed for acquiring the frames is illustrated. Third, in Section 3.3 the drone flight for the frames acquisition is described and in Section 3.4 some samples of the two datasets are shown. Finally, Section 3.5, focuses on reviewing the JSON files provided by the company, which contain the measurements of the sensors.

3.1 Wind Turbines

In order to achieve successful image stitching of a target it is of key importance to identify the characteristics which distinguish the object from its surroundings. Figure 3.1 depicts the main components of a WT. To serve the purposes of this work, more importance is given to the blade. The close-up enhances its four sides: the leading edge, the trailing edge, the high-pressure side, and the low-pressure side.

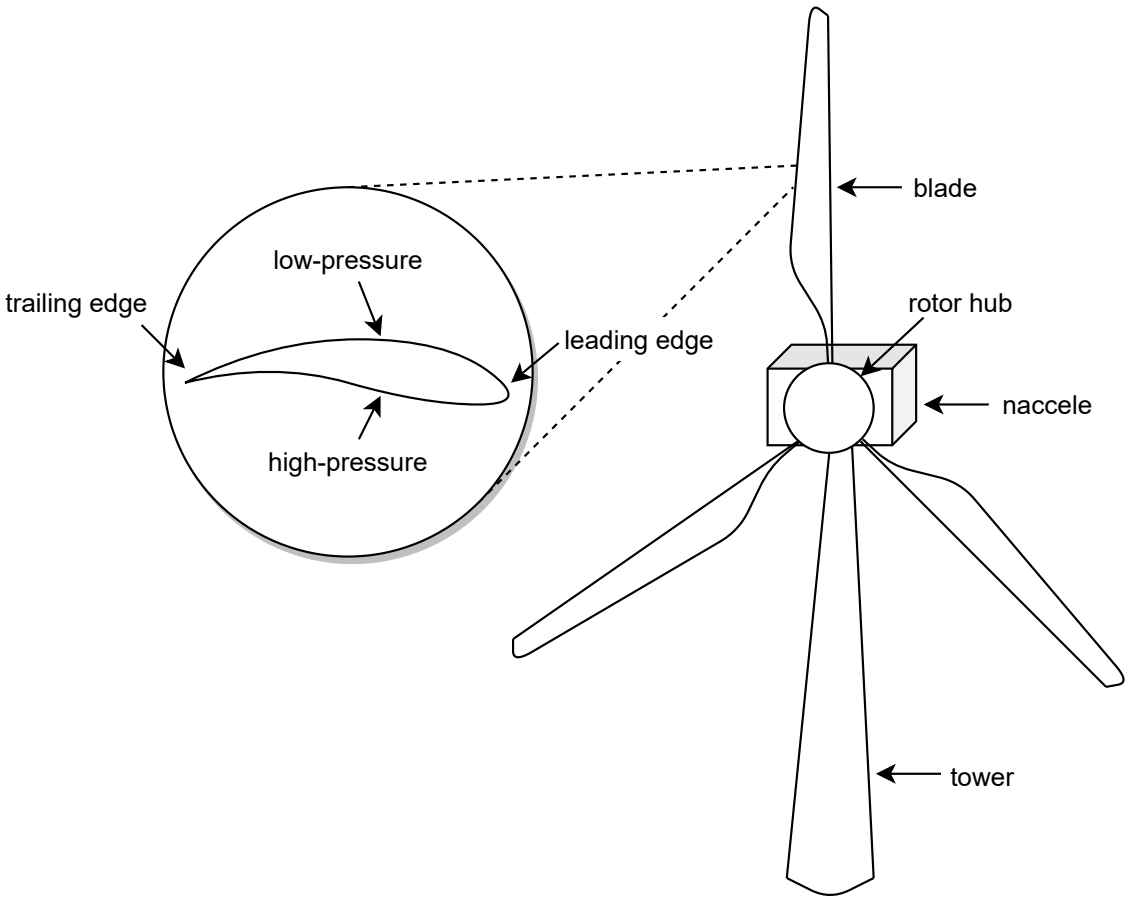


Figure 3.1: Main components of a wind turbine and a close-up on the blade highlighting its four sides from a different perspective.

The following visual properties will now be investigated: color, texture, shape, and size.

Color

Despite most being painted in white, a wind turbine can appear in any shade of gray depending on the lighting conditions [87].

Texture

The smooth surface of a wind turbine has virtually no texture [87]. When processing frames of a blade at close range, this can cause problems for feature-based image stitching algorithms.

Shape

Wind Turbine blades have a distinctive shape when compared to the surroundings in the context of wind-farms. In particular, the straight lines along the rotor blades are favorable features for edge or line detection algorithms that can be useful for background removal approaches.

Size

Over the years, WTs have grown considerably in size. This has proven to be economically profitable because as the hub-height and rotor radius increases, the average wind speed captured increases due to wind shear [14]. Nowadays, the length of a blade can go up to 107 meters. Thus, for the sake of the generalization of the project it is not recommendable to use a fixed size of the blade by any means.

3.2 Hardware setup

The UAV setup employed by the company *BladeInsight* for the imagery acquisition comprises: a camera, a Gimbal, a GPS, an IMU sensor, and a LiDAR sensor. These components are rigidly connected as depicted in Figure 3.2. Note that the camera is mounted upside down, therefore, the frames will also be upside down. Furthermore, the LiDAR sensor is placed on top of the camera, creating an offset, v, of 8 cm.

The specifications of the hardware are presented in Table 3.1.

Table 3.1: Hardware specifications.

Component type	Component name	
Drone	DJI M600 Pro	
LiDAR	Hokuyo UTM-30LX	
Camera	Sony Alpha 7RII	
	Samnyang 85 mm	
Lens	Sensor size	35.9 x 24 mm
	Pixel dimensions	7952 x 5304

Figure 3.2: Render of the setup system.

3.3 UAV imagery acquisition

A WT blade is too large for the camera's FoV. Therefore, several images are acquired along the structure. Every blade is examined at its down position and the drone flies parallel and vertically to the target, as depicted in Figure 3.3.

In order to fully cover the entire surface the procedure is repeated across the four sides of the blade, enhanced in the close-up of Figure 3.1. The UAV's flight path starts on the tip of the hight-pressure side and goes throughout the blade until the root. When the root is detected, the UAV shifts to the trailing edge side and starts following it from the root to the tip. Next, the trailing edge is examined from the tip to the root. At last, the leading edge is inspected from the root to the tip. This information is summed up at Table 3.2.

Table 3.2: Summary of the information concerning the drone flight. The blade has 4 sides, and thus 4 flight phases enumerated according to the table. The direction of the flight can be either from the tip to the root, or the other way around, from the root to the tip.

Flight Phase	Blade side	Direction
1	High-pressure	tip $\longrightarrow$ root
3	Trailing edge	tip $\longleftarrow$ root
6	Low-pressure	tip $\longrightarrow$ root
7	Leading edge	tip $\longleftarrow$ root

The quality of the data is susceptible to be influenced by external weather conditions such as wind gusts or different light exposures. A consistent capture method is necessary for success in the post-processing. Therefore, it is important that the collected data fulfills two requirements:

- Image overlap: to assure uniformity and continuity, no gaps are allowed.
- Pixel spacing: the flight path should be set to be the same distance away from the blade at all times.

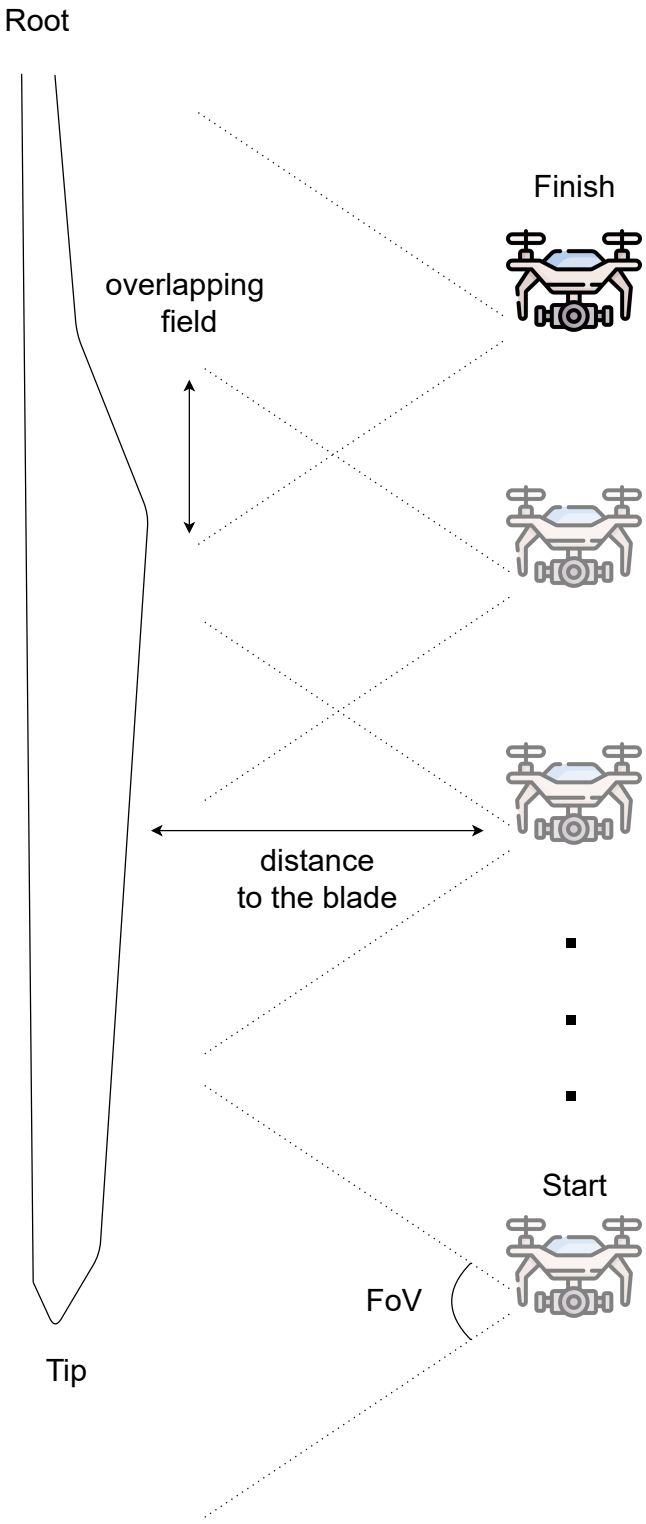
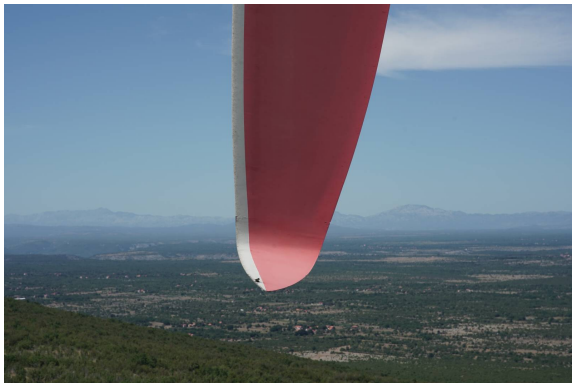


Figure 3.3: Example of a flight path performed by the drone along the blade from the tip to the root. The rays coming out of each camera represent the FoV of each frame. It is also represented the distance to the blade and the overlapping field between frames.

The acquisition procedure performed by the company *BladeInsight* ensures an overlap from 50 to 60% and a distance to the blade from 7 to 12 meters, approximately.

3.4 Dataset: clean and dirty

The results from the UAV imagery acquisition described in the previous Section are shown in Figure 3.4. The company *BladeInsight* provided two different datasets: clean and dirty . As the names suggests, the clean dataset contains a group of images acquired from a clean WT, i.e. the surface of the blades is smooth. Whereas, in the dirty dataset, it is possible to observe the texture of the surface. Figure 3.4 divides these two datasets in columns. The first column depicts 5 samples of the clean dataset which contains 190 frames in total of the 4 sides of the blade. Likewise, the second column depicts 5 samples of the dirty dataset, this time with 311 frames.



(a) DSC00002.jpg

(b) DSC00008.jpg

(c) DSC00003.jpg

(d) DSC00009.jpg

Figure 3.4: Samples of the clean (left column) and dirty (right column) datasets.

(e) DSC00004.jpg

(f) DSC00010.jpg

(g) DSC00005.jpg

(h) DSC00011.jpg

(i) DSC00006.jpg

(j) DSC00012.jpg

Figure 3.4: Samples of the clean (left column) and dirty (right column) datasets.

3.5 Sensors metadata

For every WT inspection, the company *BladeInsight*, besides providing the frames acquired by the drones, also provides two JSON files that store the data measured by the sensors. These two files are named clean_extracted_events.json and clean_extracted_lidar.json and an instance extracted from

each is presented in Listings 3.1 and 3.2, respectively.

In the instance presented in Listing 3.1 most of the variable names are self explanatory. Nonetheless, a greater relevance should be given to the drone position x, y, and z, to the distance to the airfoil center, and to the gimbal angle: roll, pitch, yaw. The drone position z along with the gimbal pitch can be useful for calculating the vertical shift between frames. In other hand, the distance to the airfoil center is helpful for normalizing the scale of the frames. The gimbal angles are further illustrated and explained in Figure 3.5.

Listing 3.1: Instance extracted from the file clean_extracted_events.json.


```
{
    "event_id": 3,
    "timestamp": 1594215117,
    "precise_timestamp": "1594215117.8136138916",
    "effective_zoom_level": 0,
    "focal_lenght": 0,
    "manual_mode": 0,
    "flight_phase": "1",
    "gps_longitude": 16.08814,
    "gps_latitude": 43.73716,
    "gps_altitude": 450.50085,
    "drone_position_x": 28.2667,
    "drone_position_y": 9.29552,
    "drone_position_z": 37.46209,
    "drone_yaw": 2.04285,
    "distance_to_airfoil_center": 6.913,
    "blade_chord": 0.06705,
    "gimbal_roll": -0.00012,
    "gimbal_pitch": 0.0008,
    "gimbal_yaw": 0,
    "image_name": "DSC00001.jpg"
}
```


Listing 3.2 presents an instance of the file clean_extracted_lidar.json, which stores the LiDAR readings. As already mentioned in Section 3.2, the LiDAR model is a Hokuyo UTM-30LX. This sensor provides an array called ”sweep” with 1080 indexes with the distance of the surrounding objects. Fur-

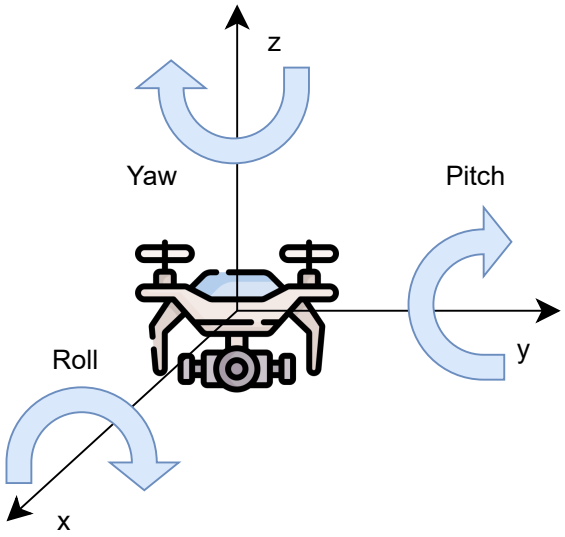


Figure 3.5: Gimbal angles: roll, pitch, and yaw. (1) Roll - Tilting camera horizon; (2) Pitch - Tilting camera up and down; (3) Yaw - Turning camera left and right.

thermore, it has a range of 270 degrees - $3/2\pi$ radians - and a maximum distance range of 30 meters ("max_range"). Everything that is within a higher distance range is reported as "inf" in the array. It is reasonable to assume that the center of the sensor is aligned with the center of the camera horizontally. Nonetheless, the sensor has a vertical offset, v, of 8 cm, as it was placed on top of the camera. Given this information, it is possible to conclude that:

- Index 540 is the center of the array, hence the projection of this point is the center of the image horizontally;
- Every index of the array has a increment of $\angle_{inc} = \frac{2/3\pi}{1080} = 0.00436$ rad - also called "angle_inc" in the JSON file. The $\angle_i$ is the angle between the optical center and the corresponding point at index i. The relation of each index i of the "sweep" array and the angle $\angle_i$ is given by the following:

$$\angle_i = \angle_{min} + \angle_{inc} \cdot i \tag{3.1}$$

  with $\angle_{min} = -2.35619$ rad $= \frac{3/2\pi}{2}$ rad - angle_min in the file.

- The background of the blade is mostly from a distance larger than 30 meters, that is read as infinite by the sensor. Therefore, to search for the blade edges indexes is to search for the first and the last non-infinite values of the array.
- The only exception to the previous statement is when the LiDAR sensor detects the tower. In this case, instead of searching for non-infinite values, one has to search for the first and the last non-

infinite consecutive values around the middle index (540) - this is explained latter on in Section 4.2.1.

Listing 3.2: Instance extracted from the file clean_extracted_lidar.json.

```
{
    "event_id": "3",
    "timestamp": 1594215117,
    "inc_angle": 0.00436,
    "angle_max": 2.35619,
    "angle_min": -2.35619,
    "max_range": 30,
    "sweep": [
        "inf",
        "inf",
        ...
        "8.79000",
        "8.79000",
        "8.75000",
        "8.67700",
        "8.61600",
        "8.53900",
        "8.47200",
        "8.46000",
        "8.42100",
        "8.36600",
        "8.35300",
        "8.27300",
        ...
        "inf",
        "inf",
        "inf"
    ]
}
```

4

Experimental Procedure

Contents

In this chapter the experimental procedure and their findings will be discussed. The first experiment is the first contact with the raw dataset. It aims to study the influence of the background on the process of stitching the blade. Therefore, the quantity, and quality of the features extracted by SIFT and ORB are analized. Ultimately, these two methods are compared. After detecting the features, they were matched employing the Euclidean Distance method. This experiment showed how distracting is the background of the frames and the importance of segmenting the blade.

The results on the first experiment lead to the second one: the blade segmentation. For such purpose, the *GraphCut* algorithm studied on section 2.4 is implemented. Moreover, the LiDAR data was required for detecting the edges of the blade. At the end of this experiment, the dataset is pre-processed and ready for image stitching.

The final experiment is about the main goal of this project: stitching the blade. First, the raw and the pre-processed dataset are given as input to a public image stitching application called *AutoStitch*. As the results were not good enough, a tailored made solution is developed. The procedure for developing this solution is explained step by step and the code implemented in *Python* is available on Github.[1]

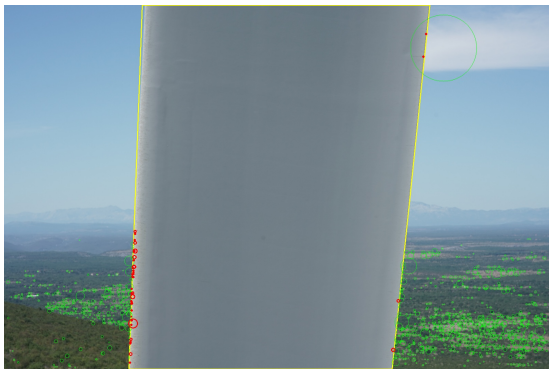
4.1 Feature-based methods with raw dataset

As mentioned in Subsection 2.2.4 plane of the background in the multiple frames is so far away from the plane of the blade that it remains almost the same. Therefore, the main goal of the first experiment - besides having a first contact with the raw dataset - is to check whether it is possible or not to stitch images without removing the background. For such purpose, the most popular methods studied in Section 2.2.3 - SIFT and ORB - were employed, in order to extract the features. The features were extracted from the clean and the dirty dataset, and then splitted into foreground and background features. The foreground features are considered the features of interest. Table 4.1 and Figure 4.1 show the results. Figure 4.1 is divided in rows and columns, where the rows depict the methods (SIFT and ORB), and the columns the sample type (clean and dirty blade). The contours of the blade are highlighted in yellow, the features detected on the surface of the blade in red, and the features detected on the background in green. Table 4.1 presents a quantitative comparison of the experiment results.

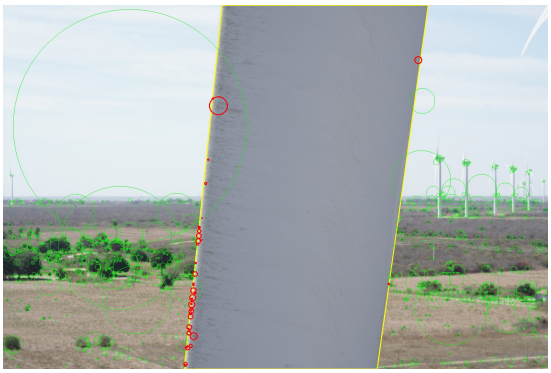
Table 4.1: Quantitative comparison of the total number of features detected and the percentage of features detected on the blade (% of interest).

method	dataset	total features	blade features	% of interest
SIFT	clean	1525	41	2.7
	dirty	1382	49	3.5
ORB	clean	500	2	0.4
	dirty	500	11	2.2

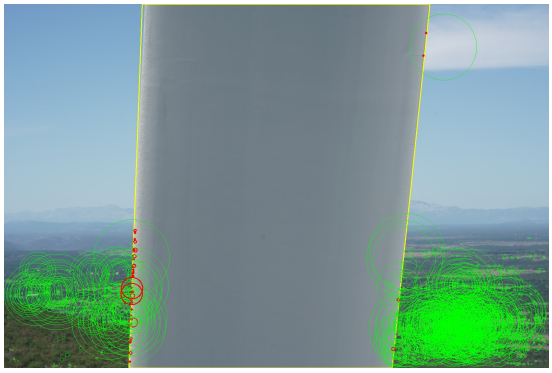
[1] https://github.com/ines18coelhoo/image-stitching-wind-turbine.git



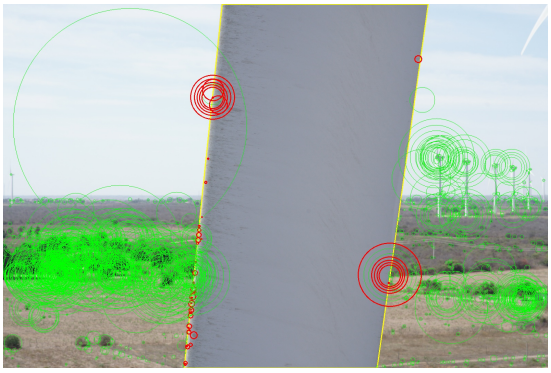
(a) Method: SIFT, Sample: clean blade.



(b) Method: SIFT, Sample: dirty blade.



(c) Method: ORB, Sample: clean blade.



(d) Method: ORB, Sample: dirty blade.

Figure 4.1: Feature detection experiment. Comparison between the features detected by the traditional feature-based algorithms SIFT and ORB on a clean and on a dirty blade. The figure is divided in rows and columns, where the rows depict the methods and the columns the sample type. Legend: blade edges (yellow); background features (green); blade features (red).

The results show that both methods are able to detect a big amount of features, albeit most being extracted from the background. The only features extracted from the surface of the blade are located on the edges. Furthermore, the results show that both algorithms are able to detect a higher number of features on the surface of the dirty blade. Additionally, the percentage of interest, i.e. the percentage of features located on the blade, are comparatively low for both methods.

After verifying that both algorithms are able to detect a relevant number of total features, the respective matches were computed. For such purpose, two consecutive frames with a mark on the surface were selected from the dirty dataset. Those frames were intentionally selected, in order to facilitate the feature-based algorithms. First the SIFT method detects the features, and second the Euclidean Distance method finds the putative matches. Figure 4.2 depicts the results.

The feature matching experiment shows, that the only match on the surface of the blade emerges on the diamond shaped mark. Although the two consecutive frames depict different parts of the blade, the background remains practically the same. Therefore, most of the matches are located on the background. Thus, the final mosaic generated will be the full overlap of the two frames. The two conducted

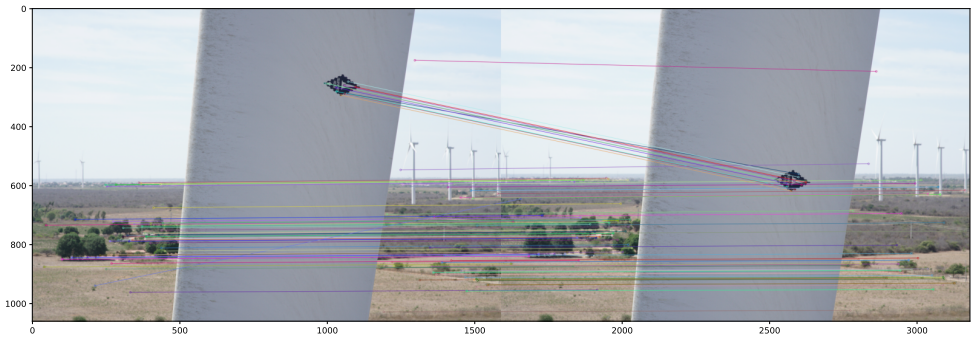


Figure 4.2: Feature matching experiment. The features were detected by SIFT algorithm and the corresponding matches were found by the Euclidean Distance method. This putative matches are highlighted in various colors.

experiments demonstrate the difficulty of traditional feature-based methods to find interest features on a WT blade, due to the distracting background. Therefore, the next Section will cover the experiments conducted on methods for subtracting the background from the frames.

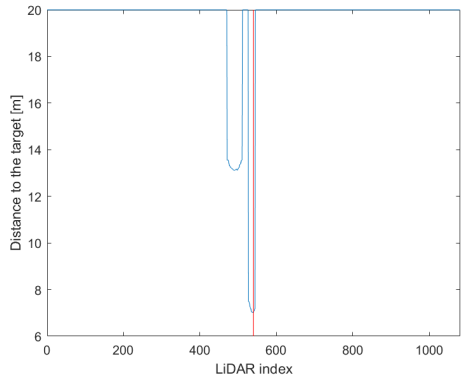
4.2 Dataset pre-processing: Blade Segmentation

This Section will describe the methods applied for image segmentation, based on the study conducted in Section 2.4. First the user input initializes the seeds of the foreground and background. However, in an effort to automatize the process and reduce the user intervention, the LiDAR data was employed to locate the blade within the frame. Therefore, the following subsection will describe how the LiDAR data was processed in order to detect the edges of the blade. Afterwards, in Subsection 4.2.2 studies how to project the LiDAR points onto the image plane. Finally, in Subsection 4.2.4, an example of the seeds placement is presented, and in Subsection 4.2.5 a geometrical correction of the segmentation is applied to the algorithm. This correction was necessary because the segmentation results showed that the algorithm was not robust against different light conditions and parts of the blade with different textures.

4.2.1 Detection of the blade edges

The main goal of this section is to find a way of locating the blade within the frame, in order to initialize the seeds of the foreground. For such purpose, the LiDAR data is required as it provides an array with 1080 indexes with the distance of the surrounding objects - see Section 3.5. An example of one of the LiDAR readings was plotted in *Matlab* and is depicted in Figure 4.3 (a). Alongside, Figure 4.3 (b) depicts

the respective frame. Note that the infinite values of the array were replaced by 20 only for visualization purposes. In the plot is possible to observe two different peaks. Since the drone tries to keep the blade centered, it is possible to conclude that it will be around index number 540 $(1081/2)$. The vertical red line represents this middle index. Thus, the peak on the left is detecting the tower and the peak on the right is detecting the blade. This could cause some confusion, as it is possible to see in Figure 4.3 (b) that the tower is on the right side. However, this phenomenon is explained by the fact that the camera is placed upside down in the setup - see section 3.2. Furthermore, it is possible to conclude that the blade is around 7 meters away, whereas the tower is around 13 meters away from the sensor.



(a) LiDAR readings.

(b) Respective frame: DSC00110.JPEG.

Figure 4.3: *Matlab* plot of the LiDAR readings, where the peak on the left represents the blade, and the peak on the right represents the tower. For visualization purposes the infinite values of the array were replaced by 20.

Computing the angle of each blade edge to the optical center using 3.1, and knowing the distance to the center of the blade it is necessary to understand how to project those LiDAR points onto an image plane. In order to do so, some concepts will be revised in the following.

4.2.2 Projecting LiDAR points to the image plane

Different digital cameras have different sensor sizes. The Sensor Size, S refers to the physical size of the sensor. The Focal Length, F, is the distance, between the optical center of a lens and the camera image sensor. Thus, it does not change, regardless of the sensor used. The FoV is the area of the inspection captured on the camera. The Angle of View (AoV) describes the angular extent of a given scene that is imaged by a camera. The Working distance, D, is the distance that separates the back of the lens from the target object [88]. Figure 4.4 illustrates the forementioned concepts. The combination of Sensor Size, S, Focal Length, F, and Working Distance, D, is related with the FoV according to (4.1).

$$\frac{\mathrm{D}}{F} = \frac{\mathrm{FoV}}{\mathrm{S}} \tag{4.1}$$

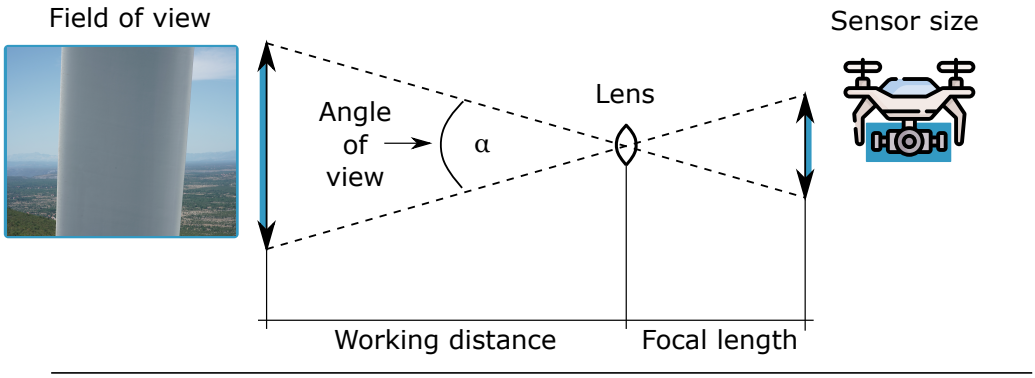


Figure 4.4: Illustration of the concepts: FoV

The Sensor Size, S, has a different width, w, and height, h, according to the information in Table 3.1. Therefore, the Focal Leght F in x and y directions has also different dimensions, according to the Pixel Size, P. The following equation solves the Focal Length in pixels:

$$F_{x,y} = \frac{F}{P_{x,y}} \tag{4.2}$$

where the Focal Length, F, is 85 mm and the Pixel Size, P, in x and y directions is given by:

$$P_{x,y} = \frac{S_{w,h}}{I_{w,h}} \tag{4.3}$$

where I stands for the image size in pixels. Finally, (4.4) projects LiDAR points onto pixel coordinates x and y:

$$\begin{cases} x = \frac{I_w}{2} - \tan(\angle_i) \cdot F_x \\ y = \frac{I_h}{2} - F_y \cdot \frac{v}{D} \end{cases} \tag{4.4}$$

in which the $\angle_i$ is the angle between the optical center and the corresponding point at index i - see (3.1) in Section 3.5 -, and v is the vertical offset between the camera and the LiDAR sensor.

4.2.3 Camera-LiDAR calibration

Initially it was assumed that the camera's optical center was vertically aligned with the LiDAR's center. Experiments were conducted in order to confirm this assumption. The angular errors between the LiDAR readings and the actual frame were measured manually and analyzed over the distance to the blade. For such purpose, a user interface was created to identify on each frame the location of both edges: left and right. The angular error is the difference between these values and the LiDAR points detecting the left and right edge, respectively. The results are depicted in Figure 4.5, where no relation could be found among the variables in study. Nonetheless, it is possible to conclude that the maximum deviation is of approximately 0.04 rad. This small deviation corresponds to an error of 131 pixels, applying (4.4). Thus,

an offset, a, of 0.04 rad was applied to $\angle_i$ in (4.4), according to the following:

$$\begin{cases} x = \frac{I_w}{2} - \tan(\angle_i + a) \cdot F_x \\ y = \frac{I_h}{2} - F_y \cdot \frac{v}{D} \end{cases} \tag{4.5}$$

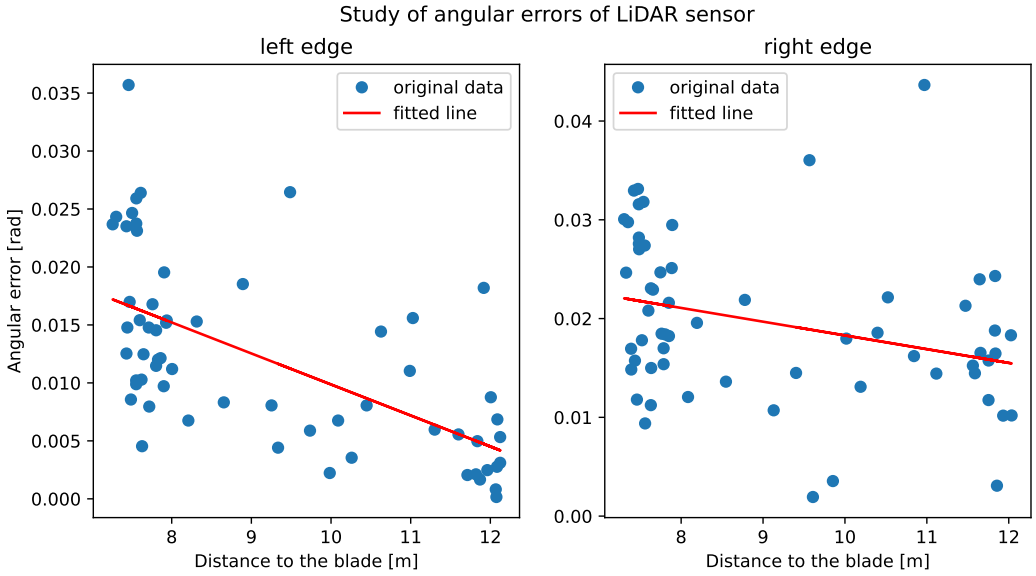
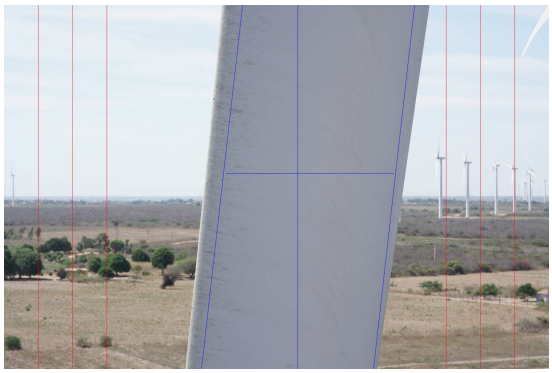


Figure 4.5: Study of the angular error of the LiDAR sensor readings over the distance to the blade measured.

4.2.4 Results: seeds placement

After detecting the edges of the blade, the seeds of both background and foreground can be placed. Figure 4.6 depicts the results. The more representative is the number of pixels selected for the seeds, the more accurate is the *GraphCut* algorithm, as described in Section 2.4. Therefore, the foreground seeds comprise: (1) a line that connects the left edge of the blade with the right edge, (2) a vertical middle line from the bottom to the top, (3) two tilted lines from the bottom to the top crossing the edges of the blade. A single vertical line from the bottom to the top on each side of the blade was enough, in most cases, to represent the background. However, in some cases such as the one depicted in Figure 4.6a the small WTs on the background were misunderstood as foreground. Therefore, more vertical lines were added in order to get a more representative portion of the background.

In Figure 4.6b the special case of the tip is depicted. First, it is necessary to detect the tip. For such purpose, the drone position is required: when the drone altitude is at its lowest point, the tip is detected. In this case, the foreground seeds are different: (2) a vertical line from the middle to the top, (3) two tilted lines connecting the edges to the top, (4) two lines connecting the edges to the tip. The background seeds follow the same rules as in the regular part of the blade. After locating the blade within the frame, it is possible to initialize the seeds for the algorithm *GraphCut* - Section 2.4.



(a) Blade.

(b) Tip of the blade.

Figure 4.6: Examples of the seeds placement of the foreground (blue) and the background (red).

4.2.5 Improving Segmentation

The results showed that this method was not robust against different light conditions or when it could not distinguish the blade from the background clouds. Figure 4.7 presents an example of a segmented frame, where it is noticeable in (a) the error in the left edge of the blade, and in (b) the improved result. The correction was implemented by simply generating a continuous line from the bottom to the top using the lower part of the left edge.

(a) Example of one frame before applying the correction.

(b) Example of the same frame after applying the correction.

Figure 4.7: Example of the results of the *GraphCut* algorithm before and after applying the correction.

Finally, the most challenging side of the blade to segment is the trailing edge. This is the thinnest side, and also the one with the largest LiDAR readings' errors. Therefore, it is not possible to generate accurate foreground seeds for the algorithm. For this case, the user interface is required to supply the seeds.

Note that the time consumption of this method is directly dependent on the number of pixels of the frames. Hence, for trial purposes, the frames were down-scaled by 90%. Nevertheless, for obtaining the

final results the images have the regular size because high-resolution is required for the following steps. After subtracting the background from the images, the dataset is ready for image stitching algorithms.

4.3 Image Stitching

The goal of this section is to carry out image Stitching to create a panorama using the overlapping images of each side of the blade. For such purpose, automatic feature extraction and computation of transformation models between pairs of images is carried out. The major steps involved in the work are as follows.

1. Extract interest points from the sequence of overlapping photos of the scene. The SIFT algorithm is used for this purpose.

2. Determine the correspondences between the interest points - Euclidean Distance.

3. For each of the pairs of images, use RANSAC algorithm for outlier rejection in the correspondences. Remove the inliers detected at the edge of the blade - further explained in Subsection 4.3.2.

4. In case the number of inliers is lower than the necessary to compute the Scaled Euclidean (< 2) apply the CLAHE filter iteratively until you get the required number of inliers.

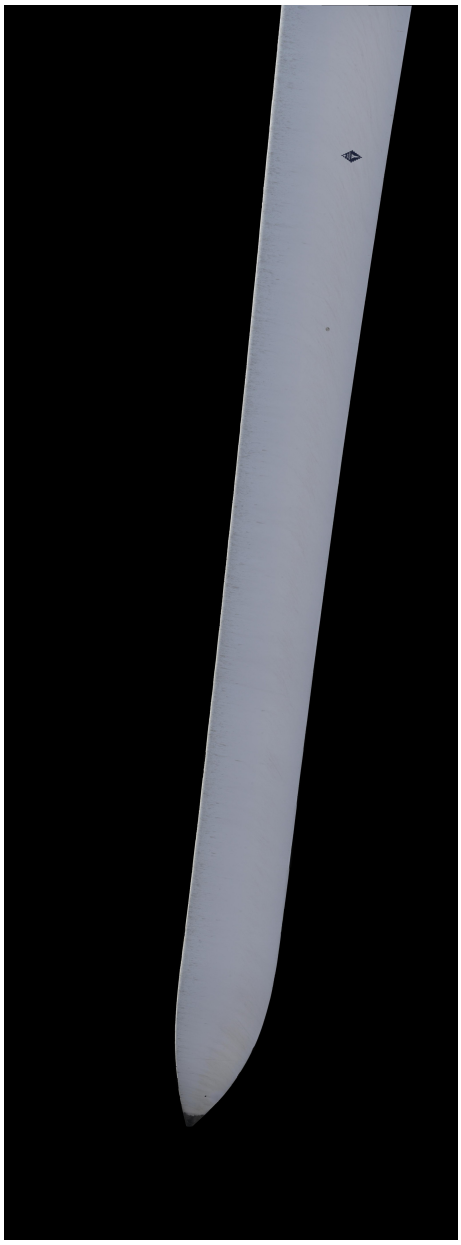
5. Compute the Scaled Euclidean Transformation between the pair of images using *opencv* function *cv2.estimateAffinePartial2D()*.

6. Stitch together the sequence of overlapping photos of the scene using the Transformation models computed to create a single panoramic photo.

4.3.1 AutoStitch

First, it is confirmed whether or not the existing commercial software can solve the problem. Thus, both datasets - with and without background - were given as input to the demo version of AutoStitch created by Brown et al. [4]. The results are depicted in Figure 4.8. In 4.8a the background is recognizable, however the blade is not. This result is expected as the background is almost the same for every frame in the dataset. From the camera's perspective the plane of the background is at an infinite distance compared to the plane of the blade. In the other hand, in 4.8b, without the background, the AutoStich [4] Application was able to recognize the blade and to stitch it. However, the demo version of AutoStitch does not support planar stitching. It assumes that the camera is rotating about a point, so distortions will be visible when stitching multiple views of a planar surface, such as the blade.

(a) Dataset with background.

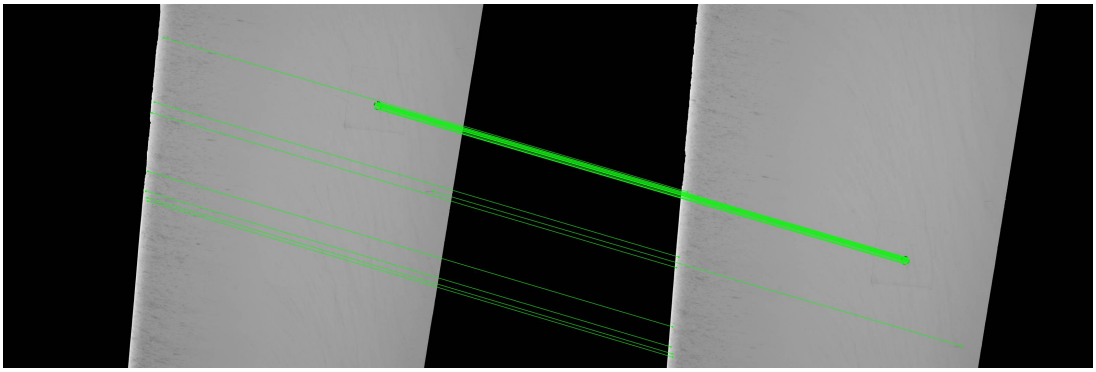


(b) Dataset without background.

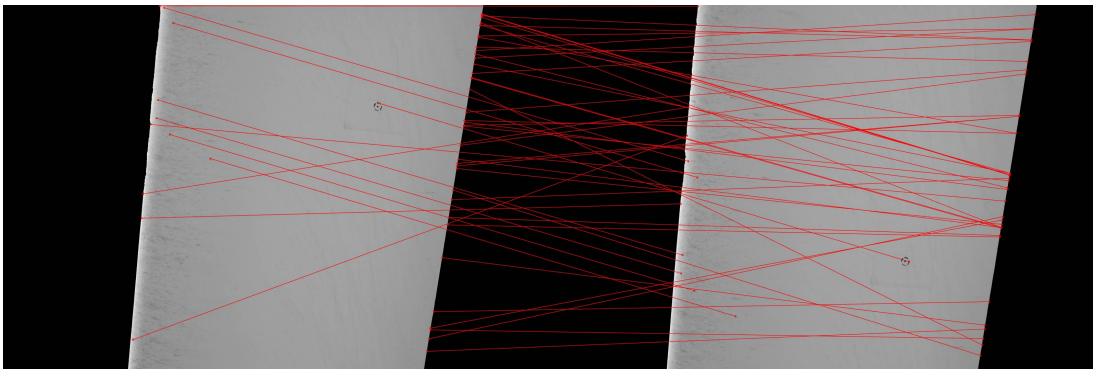
Figure 4.8: Results achieved by the demo version of AutoStitch [4].

4.3.2 Feature extraction & Feature Matching

At this stage of the process, the *OpenCV* library is used in python to conduct the feature extraction. For each of the images in the sequence, 10000 interest points are extracted using the SIFT [61] algorithm. As mentioned in Section 2.2.3.C, the matching between features from two different images is accomplished using the Euclidean Distance of the feature descriptors. However, using just the Euclidean Distance can lead to the appearance of false correspondences, specially in images with repetitive patterns. In this case, features around repetitive patterns will most likely match with more than one feature in the other image. Such matches are considered outliers and need to be eliminated. Therefore, the ratio of the Euclidean Distance of a feature with its closest and second closest match in the other image (γ) is used as a threshold. After fine tuning this parameter, all matched feature pairs between images with this ration lesser than or equal to $\gamma = 0.8$ are considered as valid correspondences between the pair of images. For debugging purposes, the resulting inliers and outliers for every image pair were stored. Figure 4.9 depicts an example of these results. Note that most of the inliers come from the circle shaped mark, and from the texture on the left edge of the blade. In other hand, most of the outliers come from edges of the blade, where it is dependent on the accuracy of the segmentation.



(a) Correspondence inliers.



(b) Correspondence outliers.

Figure 4.9: Sample of the multiple inliers and outliers correspondences stored for each pair of images.

4.3.3 Image enhancement - Contrast Limited Adaptive Histogram Equalization (CLAHE) vs Histogram Equalization

As predicted before, in some cases, the number of correspondences found is lower than the required for creating a Scaled Euclidean Transformation (< 2) between images. Therefore, image enhancement techniques are necessary to enhance the texture of the surface of the blade. In Section 2.3 was presented a brief description on the following two algorithms: Histogram Equalization and CLAHE. Figure 4.10 shows how both CLAHE and Histogram Equalization work on the same image, visualizing both the resulting image and the resulting histogram. It is possible to observe that the texture of the blade is enhanced when applying CLAHE, whereas Histogram Equalization results in dark regions. Given the results, CLAHE is preferred over Histogram Equalization.

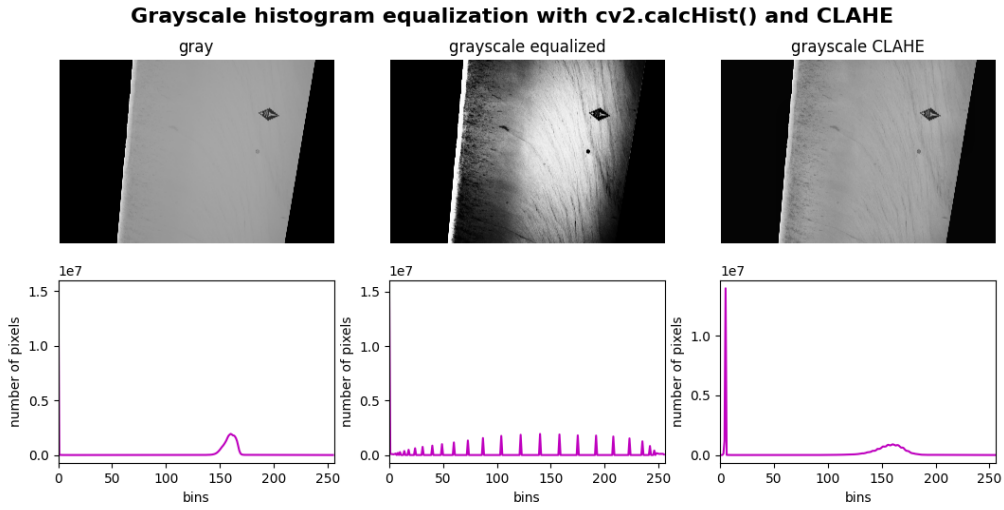
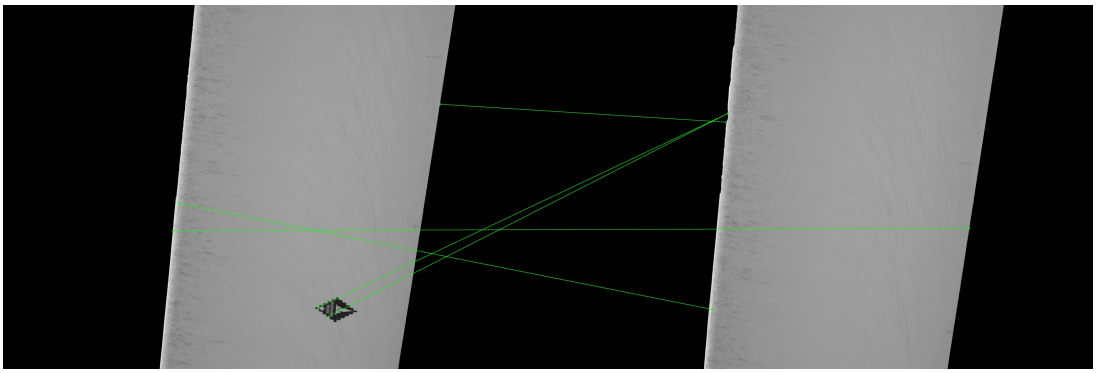


Figure 4.10: Comparison of the histograms of the original gray image, the image grayscale equalized and the image grayscale after applying CLAHE.

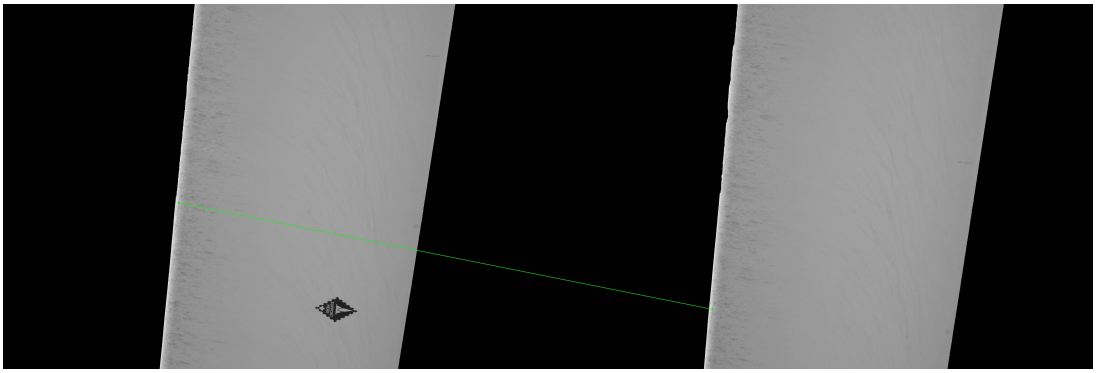
4.3.4 Improving Feature Matching

The accuracy of the feature-matching stage of the process, is dependent on the percentage of outliers among the correspondences found. Figure 4.11 depicts the inliers found throughout the process. First, notice that the images on the left (DSC00023.JPG) depicts a part of the blade exactly below to the images on the right (DSC00024.JPG). Then, notice that to identify an outlier, it is necessary to look at the slope of the lines. For a visual assessment, the lines with a positive slope and the ones with a negative slope different than the majority are outliers. In (a) 5 inliers were detected by the feature matching algorithm. However, according to the aforementioned ways of detecting an outlier it is possible to conclude that 4 of them are, in fact, outliers. The regions of the blade near to the edges are rich in features. Imperfections on the segmentation of the blade, lead to false correspondences on these areas.

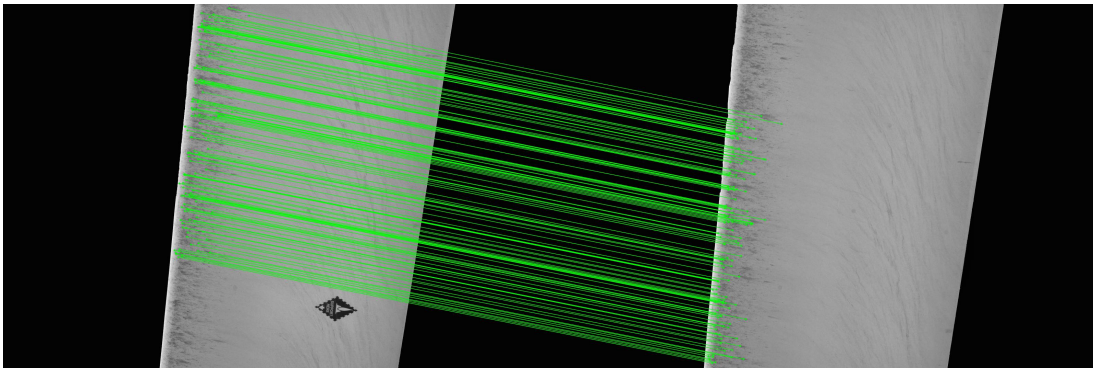
Thus, **all the matches in the edges of the blade are removed**, as shown in (b). However, only 1 inlier is not enough to compute the Scaled Euclidean Transformation (< 2). Therefore, an iterative method is implemented employing the CLAHE algorithm, as shown in (c). As the number of inliers increased from 1, to 141 it is possible to infer that this method helps finding correspondences for low texture images.



(a) Original result of feature matching. Number of inliers: 5.



(b) Removal of the matches in the edges of the blade. Number of inliers: 1.



(c) Final result after removing the matches from the edges of the blade and applying the clahe filter. Number of inliers: 141.

Figure 4.11: Improvements of the feature matching stage given the example of images DSC00023.JPG and DSC00024.JPG.

When the number of correspondences found is not sufficient to compute the Scaled Euclidean Transformation (< 2), the CLAHE algorithm is applied iteratively. That is, the parameters of the CLAHE

method - *clipLimit* and *tileGridSize* - are incremented, until the number of correspondences is enough to compute the transformation model. Figure 4.12 presents the differences in the texture of the blade when these parameters are incremented.

(a) Original. **(b)** cl=2; tgs=(8,8). **(c)** cl=4; tgs=(12,12). **(d)** cl=6; tgs=(16,16).

Figure 4.12: Iterative increment of the *clipLimit* and the *tileGridSize* when applying CLAHE.

4.3.5 Estimating Scaled Euclidean Transformation

The accuracy of the Scaled Euclidean Transformation leverages from the reliability of the inlier set. In the previous step it was ensured that the inlier set is reliable. For this step, it was used the function *cv2.estimateAffinePartial2D()* provided by the *OpenCV*. This function computes an optimal limited Affine Transformation with 4 DoF between two 2D point sets. The *OpenCV* documentation for this function can be found here [11] and is summed up in Table 4.2.

This function requires two input arrays of the same size with the coordinates of the correspondent points between two images. Afterwards, it is able to estimate an optimal 2D Affine transformation with 4 degrees of freedom limited to combinations of translation, rotation, and uniform scaling. Uses the selected algorithm for robust estimation: RANSAC or LMedS. Note that the RANSAC method can handle practically any ratio of outliers but need a threshold to distinguish inliers from outliers. The method LMedS does not need any threshold but it works correctly only when there are more than 50% of inliers. The computed transformation is then refined further (using only inliers) with the Levenberg-Marquardt method to reduce the re-projection error even more. The estimated transformation matrix is:

$$H = \begin{bmatrix} \cos\theta \cdot s & -\sin\theta & t_x \\ \sin\theta & \cos\theta \cdot s & t_y \end{bmatrix} \tag{4.6}$$

where θ is the rotation angle, s the scaling factor and t_x, t_y are translations in x, y axes respectively.

4.3.6 Edge Alignment

After applying the Scaled Euclidean Transformation and projecting the images to canvas, in very few parts of the blade it was detected that the right side was not aligned. This misalignment appears as a result of the combination of two factors: few correspondences, and almost colinear correspondences,

Table 4.2: Input and Output parameters of the *OpenCV* function cv2.estimateAffinePartial2D() with a brief description [11].

cv2.estimateAffinePartial2D()			
input	required	from	2D points (x,y) from img 1.
		to	2D points (x,y) from img 2.
	optional	method	• RANSAC - RANSAC-based robust method (default); • Least-Median of Squares (LMedS);
		ransacReproj Threshold	Maximum reprojection error in the RANSAC algorithm to consider a point as an inlier. Applies only to RANSAC.
		maxIters	The maximum number of robust methods iterations.
		confidence	Confidence level, between 0 and 1, for the estimated transformation.
		refineIters	Maximum number of iterations of Levenberg-Marquardt algorithm. Passing 0 disables refinement.
output		transformation matrix	Output 2D Affine Transformation matrix (2x3) with 4 DoF or empty matrix if the transformation could not be estimated.
		inliers	Output vector indicating which points are inliers

or too close to each other. In order to correct this limitation, a tailored algorithm called Edge Alignment was implemented. Figure 4.13 depicts the pipeline of the algorithm, which can be defined as follows:

1. Find the lower points, A and B, that limit the edges of the blade of $I2$.

2. Using the inverse of H_{12} compute points A' and B'.

$$A' = H_{12}^{-1} A$$
$$B' = H_{12}^{-1} B$$

3. Find the points, C' and D', that limit the right side of the blade in $I1$. Note that this step is not as straight forward as step 1. Some parts of the blade are curved, such as the tip and the parts

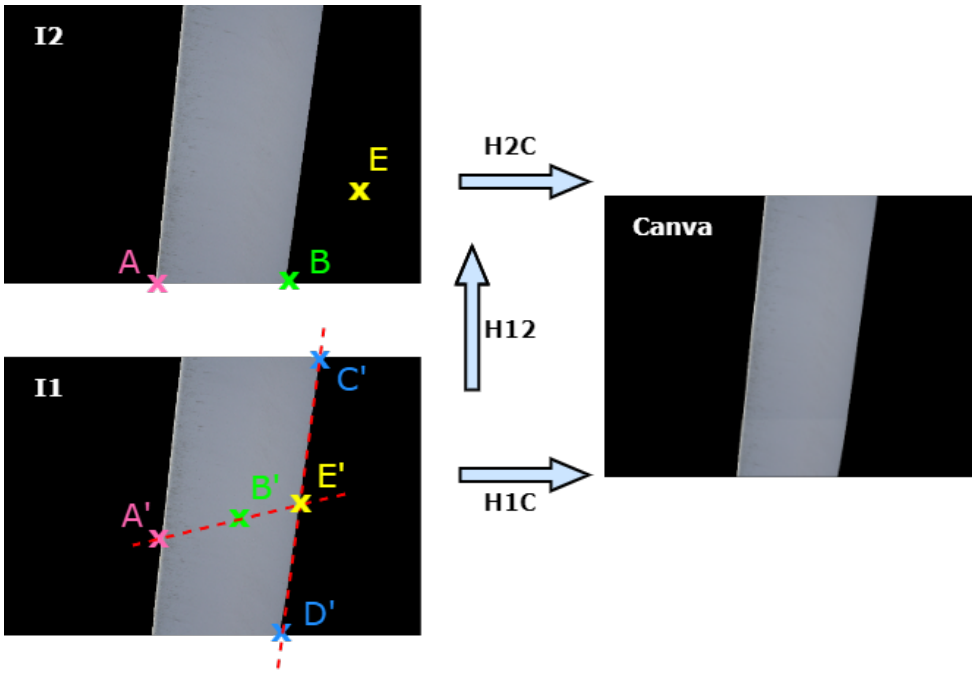


Figure 4.13: Illustration of the key points on the rescaling of the frames in order to align the right edge of the blade.

closer to the root. For this special cases, a straight line is not suitable. Thus, instead of 2 points, 4 points are searched at the edge of the blade, and a line is defined fitting those points:

$$y = a \cdot x^2 + b \cdot x + c$$

4. Define a line that connects points A' and B' and another that connects C' and D' using equation (for typical cases):

$$y = m \cdot x + b$$

5. Compute point E', that is the intersection of these two lines.

6. Compute E using the transformation H_{12}.

$$E = H_{12}E'$$

Point E is where point B should be located. Thus, if they are already at the same place the following steps are skipped.

7. Compute H_{trans_res} that transforms $I2$ in order to place point B on top of point E and keeps point

A at the exact same position.

$$A = H_{trans\_res}A$$
$$E = H_{trans\_res}B$$

H_{trans_res} is the result of a rescale and translation operation given by:

$$H_{trans\_res} = \begin{bmatrix} s & 0 & tx \\ 0 & s & ty \\ 0 & 0 & 1 \end{bmatrix}$$

where s is the factor of scale (note that it is the same for both x and y coordinates), tx and ty - the x and y translation, respectively. The resulting system of equations is given by:

$$\overset{M}{\begin{bmatrix} 1 & 0 & A_x \\ 0 & 1 & A_y \\ 1 & 0 & B_x \\ 0 & 1 & B_y \end{bmatrix}} \cdot \overset{\theta}{\begin{bmatrix} tx \\ ty \\ s \end{bmatrix}} = \overset{y}{\begin{bmatrix} A_x \\ A_y \\ E_x \\ E_y \end{bmatrix}}$$

since this matrix is not invertible it can be solved by the Pseudo-Inverse formula:

$$\theta = (M^T M)^{-1} \cdot M^T y$$

where $\theta = (tx, ty, s)$.

8. Compute H_{2C} that projects $I2$ to Canvas. H_{1C} corresponds to the indentity matrix.

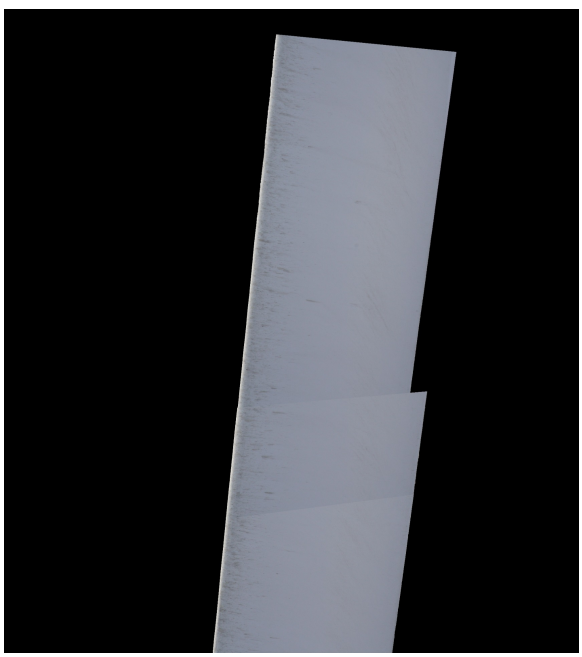
$$H_{2C} = H_{21} \cdot H_{trans\_res}$$

Figure 4.14 depicts an example of the before and after applying this algorithm.

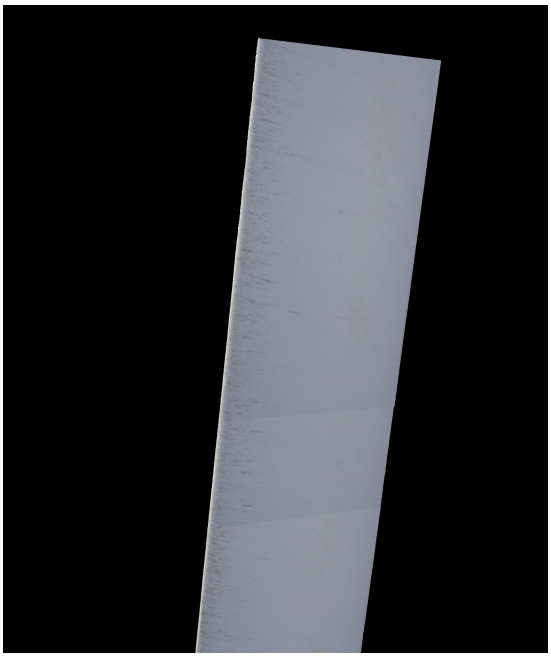
4.3.7 Panorama generation

Now that all the Scaled Euclidean Transformation matrices, $E_{i,j}$, between consecutive images have been computed, it is possible to project them onto a common reference plane called *Canvas*. The program is coded to work for any number of input images. The steps and premises involved are summarized bellow:

- All the images are projected onto the plane of the first image of the input set, which is also called the reference frame. Therefore, the transformation matrix of the first matrix is the Identity multiplied



(a) Result before applying Edge Alignment.



(b) Result after applying Edge Alignment.

Figure 4.14: Results before and after applying the Edge Alignment algorithm.

by a scale factor of 0.1, in order to downscale the dataset by 90%:

$$E_0 = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- The feature-based methods aforementioned provide a first estimation of the transformations between consecutive frames, i.e.: E_{12}, E_{23}, ... $E_{n-1,n}$. Being n the number of frames in the input set.

- The Transformation matrices from all the images to the reference image are computed as follows:

$$E_{31} = E_{12}^{-1} \cdot E_{23}^{-1}$$
$$E_{41} = E_{34}^{-1} \cdot E_{31}$$
$$...$$
$$E_{n,1} = E_{n-1,n}^{-1} \cdot E_{n-1,1}$$

- Every Transformation matrix is adjusted applying the tailored made algorithm explain in subsection 4.3.6.

- Now, all the transformation models need to be left-multiplied by a translation matrix, E_t, so as to

translate the first image (reference) to the bottom of the canvas. This translation matrix is of the form:

$$E_t = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix}$$

where tx and ty are the maximum values of translation obtained in the other Transformation matrices. That is the same as the translation parameters of Scaled Euclidean Transformation E_{n1}.

- The size of the canvas is computed as follows. The length of the canvas corresponds to the length of every image added to the largest translation in x. Likewise, the height of the canvas is the height of every frame added to the largest translation in y.
- Everything is set for projecting the input set to canvas.

5

Results

Contents

In the previous Chapter, it was described the whole tailored made method and procedure for building the panoramic view of a WT blade. In this Chapter, the final result depicted in Figure 5.1 is evaluated quantitative - Section 5.1 - and qualitatively - Section 5.2. Besides, the performance of the existing commercial software AutoStitch [4] is compared with the tailored made solution - Section 5.3.

5.1 Quantitative Evaluation

Since the final result is an image, it is difficult to come up with a number to evaluate quantitatively the overall quality of the mosaic. In other applications, to perform a quantitative evaluation it is required a ground-truth to be compared with the results. In this case, this ground-truth could be, for instance, an image of the whole blade. However, this is not available. Therefore, evaluating quantitatively the accuracy of the stitching becomes an hard task.

As an alternative, is computed a numerical comparison between the physical length of the blade and the length of the blade in the mosaic, both in meters. For such purpose, (4.2) and (4.3) are combined in order to calculate the Focal Length in pixels only in the y direction, F_y:

$$F_y = \frac{F \cdot I_h}{S_h} \tag{5.1}$$

where the Focal Length, F, is 85 mm, the Image size in pixels is $0.1 \cdot 5304$ pixels due to the fact that the images were downsized by 90%, and the Sensor height, S_h, is 24mm. Thus, the Focal Length in pixels, F_y, is 1878.5. Having the Focal Length in pixels, and using the pinhole camera model traduced by (5.2), it is possible to compute the size of a pixel in meters, Y, on the plane of the blade:

$$y = F_y \cdot \frac{Y}{D} \Rightarrow Y = \frac{D}{F_y} \tag{5.2}$$

where D is the Working Distance, Y is the size of a pixel in meters on the plane of the blade, and y is one pixel on the image plane. Despite the efforts of the drone, the distance to the blade varies along the flying path. Nonetheless, it should only be considered the distance to the blade of the first frame. This is due to the fact that the first frame is designed the anchor, i.e. all the other frames are projected into the plane of the first frame. Therefore, accessing file *dirty_extracted_events.JSON*, the Working Distance, D, corresponds to 7.471 meters. According to (5.2), the size of each pixel in meters is 0.003977 - $\frac{7.471}{1878.5}$. Now, the only thing that is left to do is to multiply this number by the number of pixels of the length of the blade in Figure 5.1 - 11665. Finally, the computed size of the blade is 46.393 meters, whereas the real size of the blade 48.7 meters. The relative error is given by:

$$\epsilon_{\%} = \frac{|46.4 - 48.7|}{48.7} \cdot 100 = 4.7\% \tag{5.3}$$

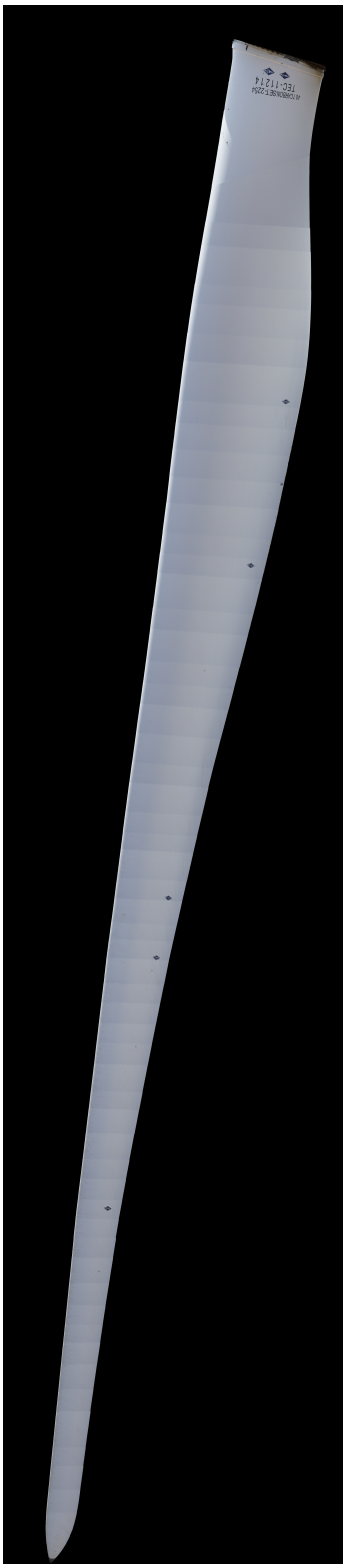


Figure 5.1: Final result: panoramic view of the low-pressure side of a WT blade.

Another relevant quantitative evaluation, regarding the computational efficiency, is the computational time and the memory usage. This software is, as expected, highly dependent on the number of images in the dataset as well as the number of features in each image. Therefore, the higher the number of images the longer it will take to produce the final mosaic. Moreover, the more information the images contain (more texture), the higher the number of features, and this can also influence the time consumption of the algorithm. Another issue is the lack of memory, especially in lower-end computers, or when one uses datasets with very high-resolution images. These problems can be tackled using smaller datasets or reducing the resolution of the input images. In this case, using lower resolution images is not an option, because this would decrease the performance of the feature detectors. Thus, only after detecting the features for each frame, the images were downsized by 90% in order to decrease the computational time. The final result depicted in Figure 5.1 contains 59 frames with 795x530 pixels. The software took around 30 seconds per image, that is approximately 30 minutes in total.

5.2 Qualitative Evaluation

Figure 5.1 depicts the final result using as input the images from the low-pressure side. Assessing visually the result and zooming it in, it is possible to observe that the edges of the blade almost do not have discontinuities. Furthermore, the lines separating images are mostly parallel, which was the objective, as the drone tries to keep the same roll angle during the flight - see Figure 3.5. These lines are also called seams, in the vocabulary of image stitching, and they are visible contrary to what most of image stitching algorithms do. This behavior was expected as no blending algorithm is applied. The purpose of not applying any blending algorithm is to get a clear vision of the defects on the blade. The application of blending algorithms could cause some blurring around the regions of the defects.

Nevertheless, the part closer to the root of the blade seems to be larger than the part closer to the tip. Analyzing the dataset, in frames correspondent to the part closer to the tip, the blade fills approximately $1/4$ of the entire frame - Figure 5.2a. In other hand, in the frames correspondent to the part closer to the root, the blade occupies about $3/4$ of the entire frame - Figure 5.2b. This is one of the factors that helps explaining this phenomenon. In other hand, analyzing file *dirty_extracted_events.JSON* the drone is actually moving away from the blade in regions closer to the root. Which means that this part of the blade is wider. Having in mind that the projection is planar and the root of the blade is twisted in relation to the tip, this is the expected result. The planar projection also causes the distortion observed at the final frame of the root.

(a) Sample of an image closer to the tip - DSC00010.jpg.

(b) Sample of an image closer to the root - DSC00056.jpg.

Figure 5.2: Two samples that illustrate the different portions of the frames occupied by the blade. The frames closer to the tip occupy a smaller portion of the frame, than the frames closer to the root.

5.3 Comparison with commercial software

In this Section, the performance of the existing commercial software is compared with the tailored made solution. For such purpose, the demo version of AutoStitch created by Brown et al. [4] is downloaded. After downloading the software, the same dataset given as input to the tailored made solution is given to AutoStitch. The result is depicted in Figure 5.3. The AutoStich [4] Application was not able to build the mosaic of the blade. This could be explained by the fact that the demo version of AutoStitch does not support planar stitching. It assumes that the camera is rotating about a point, so distortions will be visible when stitching multiple views of a planar surface, such as a blade. Furthermore, this application does not let the user define the order of the frames to stitch, which could make the process simpler and the result more accurate. This software was more successful for smaller portions of the blade, as seen before in Section 4.3.1 - Figure 4.8b.

Figure 5.3: Autostitch software result having as input the entire dataset of the low-pressure side of the blade. The same input given to the tailored made solution.

6

Conclusions & Future Work

Contents

This Chapter will conclude the study by summarizing the key research findings. It will answer the research aims and questions, as well as discuss the value and contribution of this study. Finally, it will also review the limitations of the work and propose opportunities for future research.

6.1 Conclusions

The main goal of the project is to investigate, develop and implement an end-to-end system to automatically produce a mosaic, of a WT blade. For this purpose, the system developed takes as input the multitude of images acquired by the drone during the inspection and merges them into a single one. The process is fully automated, i.e. it does not require any manual operations. The developing procedure faced two major challenges: the distracting background of a WT farm environment and the featureless surface of the blade.

For tackling the first challenge, the algorithm *GraphCut* was applied. This method requires the initialization of seeds both in the foreground, and in the background of the blade. In order to locate the blade within the frame, the LiDAR data was used. The results showed that this method was not robust against different light conditions or when it could not distinguish the blade from the background clouds. Therefore, a geometrical correction had to be applied by simply generating a continuous straight line on the edge of the blade. For tackling the second challenge, an iterative image enhancement technique was developed. The Contrast Limited Adaptive Histogram Equalization (CLAHE) parameters were increased iteratively until the feature matching algorithm could find the minimum required number of correspondences. Still in the feature matching stage, it was discovered that most of the inliers located on the edges of the blade were actually outliers. Thus, every matching pair located on the edges of the blade has been removed.

The image stitching pipeline comprehended three stages: feature extraction, feature matching, and transformation model estimation. For the feature extraction, the experiments indicate that the SIFT method is the most robust. The feature matching task could have been done in various ways, but the most accepted one ended up being the Euclidean distance between the feature descriptors, whereas, the RANSAC method filtered the inliers. Regarding the final stage, the transformation model estimation, the Scaled Euclidean transformation is the proper transformation, since the Euclidean proved to be insufficient, and the Affine Transformation an overkill.

The edge alignment of the blade is of most importance for the success of the stitching, thus a method was develop to ensure that the edges were consistently aligned. The left side of the blade is rich in texture, therefore the feature-based methods alone were enough to ensure the correct alignment of this side. In other hand, the right edge of the blade was misaligned sometimes. The tailored made algorithm for tackling this challenge actuates after the estimation of the model transformation - the final step of the stitching pipeline. At this point, the images are already in the same coordinate system, and the

algorithm checks pair by pair if the edges are correctly aligned. In case they are not, it adjusts the model transformation previously calculated.

All in all, the result achieved looks seamless in geometry, i.e. there are no discontinuities between frames, specially on the edges of the blade. In contrast with the result obtained with the commercial available software AutoStitch, where the blade is not even perceptible.

6.2 System Limitations and Future Work

In the first experiments with the raw dataset, it was soon perceptible the importance of removing the background from the images. In order to get a successful stitching of the blade, the background was too distracting and had to be subtracted. Therefore, Image Segmentation Techniques were studied and implemented. The blade segmentation algorithm created has some limitations related to the fact that it deeply relies on the IMU and LiDAR data. However, this data acquired by the sensors revealed to be unreliable. The success of the GraphCut Algorithm is dependent on the initialization of the foreground and background seeds placement. If, for instance, one foreground seed is misplaced at the background, the segmentation will be erroneous. The leading edge, and the trailing edge are the sides of the blade that present worse results. These two sides are the most affected ones by the LiDAR inaccurate readings, since they are the thinnest.

There might be two explanations for the unreliability of the LiDAR data. The first one is that, despite the efforts, the camera and the LiDAR sensor are not rigidly aligned. Even between UAV acquisitions, a minimal change in the setup might cause a huge deviation in the readings. The second reason is that the accuracy of locating the points at the edges of the blade is susceptible to a high level of noise, both in distance and in orientation.

Regarding the image stitching pipeline, the attempt for creating a mosaic using the clean dataset was not well succeeded, since there were not enough features. Therefore, a new approach combining the feature-based method already created with a direct projection using the IMU data was experimented. The idea was to try to apply first the feature-based methods, and only in the cases where the features were sparse, to use a direct projection. However, once again, the accuracy of the IMU data was not enough to produce good results.

Due to the aforementioned reasons, it is suggested the creation of a calibration algorithm that calculates the offset between the camera and the LiDAR sensor. Furthermore, the employment of sensors that provide more reliable data would be an add-on. Regarding the image segmentation, methods detecting vertical lines should be tried on, such as the convolutional kernel. In this case, a different algorithm for detecting the tip of blade must be added since the lines are not perpendicular at this part of the blade.

In addition, the software developed was tested for the low-pressure side of the blade. Future work should test the algorithm with the other three sides: high-pressure, leading, and trailing edge. Regarding

the pipeline for monitoring a WT, the algorithms for defect detection can precede the stitching software, and generate a global defect map. For visualization purposes it would facilitate to locate the defect within the blade.

Bibliography

[1] Wind Europe, "Wind Energy in Europe. 2020 Statistics and the outlook for 2021-2025," last accessed 28 February 2021. [Online]. Available: https://windeurope.org/data-and-analysis/product/wind-energy-in-europe-in-2020-trends-and-statistics/

[2] Y. Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in nd images," in *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, vol. 1. IEEE, 2001, pp. 105–112.

[3] C. Rother, V. Kolmogorov, and A. Blake, "" grabcut" interactive foreground extraction using iterated graph cuts," *ACM Vransactions on Graphics (TOG)*, vol. 23, no. 3, pp. 309–314, 2004.

[4] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *International Journal of Computer Vision*, vol. 74, no. 1, pp. 59–73, 2007.

[5] B. Zitova and J. Flusser, "Image registration methods: a survey," *Image and vision computing*, vol. 21, no. 11, pp. 977–1000, 2003.

[6] S. A. K. Tareen and Z. Saleem, "A comparative analysis of sift, surf, kaze, akaze, orb, and brisk," in *2018 International conference on computing, mathematics and engineering technologies (iCoMET)*. IEEE, 2018, pp. 1–10.

[7] E. Karami, S. Prasad, and M. S. Shehata, "Image matching using sift, surf, BRIEF and ORB: performance comparison for distorted images," *CoRR*, vol. abs/1710.02726, 2017. [Online]. Available: http://arxiv.org/abs/1710.02726

[8] F. Hidalgo and T. Bräunl, "Evaluation of several feature detectors/extractors on underwater images towards vslam," *Sensors*, vol. 20, no. 15, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/15/4343

[9] D. Bojanić, K. Bartol, T. Pribanić, T. Petković, Y. D. Donoso, and J. S. Mas, "On the comparison of classic and deep keypoint detector and descriptor methods," in *2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)*. IEEE, 2019, pp. 64–69.

[10] H. Chatoux, F. Lecellier, and C. Fernandez-Maloigne, “Comparative study of descriptors with dense key points,” in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 1988–1993.

[11] Doxygen, “Camera Calibration and 3D Reconstruction,” last accessed 22 April 2022. [Online]. Available: https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#gad767faff73e9cbd8b9d92b955b50062d

[12] D. Capel, *Image Mosaicing*. London: Springer London, 2004, pp. 47–79. [Online]. Available: https://doi.org/10.1007/978-0-85729-384-8_4

[13] M. Grujicic, G. Arakere, B. Pandurangan, V. Sellappan, A. Vallejo, and M. Ozen, “Multidisciplinary design optimization for glass-fiber epoxy-matrix composite 5 mw horizontal-axis wind-turbine blades,” *Journal of Materials Engineering and Performance*, vol. 19, no. 8, pp. 1116–1127, 2010.

[14] A. Chehouri, R. Younes, A. Ilinca, and J. Perron, “Review of performance optimization techniques applied to wind turbines,” *Applied Energy*, vol. 142, pp. 361–388, 2015.

[15] Wind Europe, “Wind Energy and Economic recovery in Europe,” last accessed 28 February 2021. [Online]. Available: https://windeurope.org/data-and-analysis/product/wind-energy-and-economic-recovery-in-europe/

[16] B. Hu, W. Li, C. Song, K. Yuan, F. Zhao, and D. Wei, “Surface damage detection method for blade of wind turbine based on image segmentation,” in *2020 5th International Conference on Communication, Image and Signal Processing (CCISP)*. IEEE, 2020, pp. 154–158.

[17] H. Im and B. Kim, “Numerical study on the effect of blade surface deterioration by erosion on the performance of a large wind turbine,” *Journal of Renewable and Sustainable Energy*, vol. 11, no. 6, p. 063308, 2019.

[18] J. Ruan, S. C. M. Ho, D. Patil, and G. Song, “Structural health monitoring of wind turbine blade using piezoceremic based active sensing and impedance sensing,” in *Proceedings of the 11th IEEE International Conference on Networking, Sensing and Control*. IEEE, 2014, pp. 661–666.

[19] M. Shafiee, Z. Zhou, L. Mei, F. Dinmohammadi, J. Karama, and D. Flynn, “Unmanned aerial drones for inspection of offshore wind turbines: A mission-critical failure analysis,” *Robotics*, vol. 10, no. 1, 2021. [Online]. Available: https://www.mdpi.com/2218-6581/10/1/26

[20] O. Moolan-Feroze, K. Karachalios, D. N. Nikolaidis, and A. Calway, “Simultaneous drone localisation and wind turbine model fitting during autonomous surface inspection,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 2014–2021.

[21] P. Poozesh, A. Sabato, A. Sarrafi, C. Niezrecki, P. Avitabile, and R. Yarala, "Multicamera measurement system to evaluate the dynamic response of utility-scale wind turbine blades," *Wind Energy*, vol. 23, no. 7, pp. 1619–1639, 2020.

[22] S. Shivaram, "Structural Health Monitoring of Wind Turbine Blades using Unmanned Air Vehicles," Master's thesis, University of Dublin, Trinity College, September 2015.

[23] Q. Hongwu, S. Haixin, C. Wei, and D. Mengcong, "Structural health monitoring wtb using the effectiveness of graphical programming packages analysis on acoustic emission data," in *2015 IEEE Fifth International Conference on Big Data and Cloud Computing*. IEEE, 2015, pp. 207–212.

[24] L. Wang and Z. Zhang, "Automatic detection of wind turbine blade surface cracks based on uav-taken images," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 9, pp. 7293–7303, 2017.

[25] Y. Du, S. Zhou, X. Jing, Y. Peng, H. Wu, and N. Kwok, "Damage detection techniques for wind turbine blades: A review," *Mechanical Systems and Signal Processing*, vol. 141, p. 106445, 2020.

[26] A. Joshuva, A. K. Aslesh, and V. Sugumaran, "State of the art of structural health monitoring of wind turbines," *International Journal of Mechanical and Production Engineering Research and Development*, vol. 9, pp. 95–112, 2019.

[27] D. Willis, C. Niezrecki, D. Kuchma, E. Hines, S. Arwade, R. Barthelmie, M. DiPaola, P. Drane, C. Hansen, M. Inalpolat *et al.*, "Wind energy research: State-of-the-art and future research directions," *Renewable Energy*, vol. 125, pp. 133–154, 2018.

[28] Y. T. Eugene Chian and J. Tian, "Surface defect inspection in images using statistical patches fusion and deeply learned features," *AI*, vol. 2, no. 1, pp. 17–31, 2021. [Online]. Available: https://www.mdpi.com/2673-2688/2/1/2

[29] A. Shihavuddin, X. Chen, V. Fedorov, A. Nymark Christensen, N. Andre Brogaard Riis, K. Branner, A. Bjorholm Dahl, and R. Reinhold Paulsen, "Wind turbine surface damage detection by deep learning aided drone inspection analysis," *Energies*, vol. 12, no. 4, p. 676, 2019.

[30] J. Zhang, G. Cosma, and J. Watkins, "Image enhanced mask r-cnn: A deep learning pipeline with new evaluation measures for wind turbine blade defect detection and classification," *Journal of Imaging*, vol. 7, no. 3, p. 46, 2021.

[31] D. Sarkar and S. K. Gunturi, "Wind turbine blade structural state evaluation by hybrid object detector relying on deep learning models," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–14, 2020.

[32] E. Adel, M. Elmogy, and H. Elbakry, “Image stitching based on feature extraction techniques: a survey,” *International Journal of Computer Applications*, vol. 99, no. 6, pp. 1–8, 2014.

[33] D. Ghosh and N. Kaabouch, “A survey on image mosaicing techniques,” *Journal of Visual Communication and Image Representation*, vol. 34, pp. 1–11, 2016, elsevier.

[34] R. Szeliski, “Image alignment and stitching: A tutorial,” *Foundations and Trends® in Computer Graphics and Vision*, vol. 2, no. 1, pp. 1–104, 2006.

[35] R. Abraham and P. Simon, “Review on mosaicing techniques in image processing,” in *2013 Third International Conference on Advanced Computing and Communication Technologies (ACCT)*, 2013, pp. 63–68.

[36] H. Zhang and M. Zhao, “Panoramic image stitching using double encoder–decoders,” *SN Computer Science*, vol. 2, no. 2, pp. 1–12, 2021.

[37] Y. Xiong and K. Pulli, “Fast image stitching and editing for panorama painting on mobile phones,” in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, 2010, pp. 47–52.

[38] T. Twetman, “Multi view image stitching of planar surfaces on mobile devices,” Master’s thesis, KTH Royal School of Technology, December 2015.

[39] Q. Yang, C. Wang, Y. Gao, H. Qu, and E. Y. Chang, “Inertial sensors aided image alignment and stitching for panorama on mobile phones,” in *Proceedings of the 1st international workshop on Mobile location-based service*, 2011, pp. 21–30.

[40] S. Ait-Aoudia, R. Mahiou, H. Djebli, and E.-H. Guerrout, “Satellite and aerial image mosaicing-a comparative insight,” in *2012 16th International Conference on Information Visualisation*. IEEE, 2012, pp. 652–657.

[41] M. Bryson, A. Reid, F. Ramos, and S. Sukkarieh, “Airborne vision-based mapping and classification of large farmland environments,” *Journal of Field Robotics*, vol. 27, no. 5, pp. 632–655, 2010.

[42] L. Carozza, D. Tingdahl, F. Bosché, and L. Van Gool, “Markerless vision-based augmented reality for urban planning,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 29, no. 1, pp. 2–17, 2014.

[43] M. Quaritsch, E. Stojanovski, C. Bettstetter, G. Friedrich, H. Hellwagner, B. Rinner, M. Hofbaur, and M. Shah, “Collaborative microdrones: applications and research challenges,” in *Proceedings of the 2nd International Conference on Autonomic Computing and Communication Systems*, 2008, pp. 1–7.

[44] D. Kuang and T. Schmah, "Faim–a convnet method for unsupervised 3d medical image registration," in *International Workshop on Machine Learning in Medical Imaging*. Springer, 2019, pp. 646–654.

[45] M. Scaioni, L. Barazzetti, and M. Gianinetto, "Multi-image robust alignment of medium-resolution satellite imagery," *Remote Sensing*, vol. 10, no. 12, p. 1969, 2018.

[46] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, J.-K. Kämäräinen, M. Danelljan, L. Č. Zajc, A. Lukežič, O. Drbohlav *et al.*, "The eighth visual object tracking vot2020 challenge results," in *European Conference on Computer Vision*. Springer, 2020, pp. 547–601.

[47] K. Joshi, "A survey on real time image stitching," 2020.

[48] J. N. Sarvaiya, S. Patnaik, and S. Bombaywala, "Image registration by template matching using normalized cross-correlation," in *2009 international conference on advances in computing, control, and telecommunication technologies*. IEEE, 2009, pp. 819–822.

[49] B. S. Reddy and B. N. Chatterji, "An fft-based technique for translation, rotation, and scale-invariant image registration," *IEEE transactions on image processing*, vol. 5, no. 8, pp. 1266–1271, 1996.

[50] F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, and P. Suetens, "Multimodality image registration by maximization of mutual information," *IEEE transactions on Medical Imaging*, vol. 16, no. 2, pp. 187–198, 1997.

[51] J. Li, L. Ma, Y. Fan, N. Wang, K. Duan, Q. Han, X. Zhang, G. Su, C. Li, and L. Tang, "An image stitching method for airborne wide-swath hyperspectral imaging system equipped with multiple imagers," *Remote Sensing*, vol. 13, no. 5, p. 1001, 2021.

[52] Y. Yuan, F. Fang, and G. Zhang, "Superpixel-based seamless image stitching for uav images," *IEEE Transactions on Geoscience and Remote Sensing*, 2020.

[53] R. Xie, J. Tu, J. Yao, M. Xia, and S. Li, "A robust projection plane selection strategy for uav image stitching," *International Journal of Remote Sensing*, vol. 40, no. 8, pp. 3118–3138, 2019.

[54] Y. Zhao, Y. Cheng, X. Zhang, S. Xu, S. Bu, H. Jiang, P. Han, K. Li, and G. Wan, "Real-time orthophoto mosaicing on mobile devices for sequential aerial images with low overlap," *Remote Sensing*, vol. 12, no. 22, p. 3739, 2020.

[55] S. Bu, Y. Zhao, G. Wan, and Z. Liu, "Map2dfusion: Real-time incremental uav image mosaicing based on monocular slam," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4564–4571.

[56] Y. Zhao, G. Liu, S. Xu, S. Bu, H. Jiang, and G. Wan, "Fast georeferenced aerial image stitching with absolute rotation averaging and planar-restricted pose graph," *IEEE Transactions on Geoscience and Remote Sensing*, 2020.

[57] M. R. Saleem, J.-W. Park, J.-H. Lee, H.-J. Jung, and M. Z. Sarwar, "Instant bridge visual inspection using an unmanned aerial vehicle by image capturing and geo-tagging system and deep convolutional neural network," *Structural Health Monitoring*, p. 1475921720932384, 2020.

[58] S. Yahyanejad, D. Wischounig-Strucl, M. Quaritsch, and B. Rinner, "Incremental mosaicking of images from autonomous, small-scale uavs," in *2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance*. IEEE, 2010, pp. 329–336.

[59] M. Xia, J. Yao, R. Xie, L. Li, and W. Zhang, "Globally consistent alignment for planar mosaicking via topology analysis," *Pattern Recognition*, vol. 66, pp. 239–252, 2017.

[60] Planet Observer, "The one-stop shop for all imagery products," [Online; accessed April 13, 2021]. [Online]. Available: https://planetobserver.com/wp-content/uploads/2020/08/PlanetObserver-satellite-imagery-Denmark-Copenhagen-box-1.jpg

[61] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[62] K. G. Derpanis, "The harris corner detector," *York University*, vol. 2, 2004, citeseer.

[63] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *European conference on computer vision*. Springer, 2010, pp. 778–792.

[64] A. Alahi, R. Ortiz, and P. Vandergheynst, "Freak: Fast retina keypoint," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. Ieee, 2012, pp. 510–517.

[65] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European conference on computer vision*. Springer, 2006, pp. 430–443.

[66] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[67] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, "Kaze features," in *European conference on computer vision*. Springer, 2012, pp. 214–227.

[68] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *2011 International conference on computer vision*. Ieee, 2011, pp. 2548–2555.

[69] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.

[70] T. Lindeberg, "Scale-space theory: A basic tool for analyzing structures at different scales," *Journal of applied statistics*, vol. 21, no. 1-2, pp. 225–270, 1994.

[71] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger, "Adaptive and generic corner detection based on the accelerated segment test," in *European conference on Computer vision*. Springer, 2010, pp. 183–196.

[72] Y. Ke and R. Sukthankar, "Pca-sift: A more distinctive representation for local image descriptors," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2. IEEE, 2004, pp. II–II.

[73] R. Arandjelović and A. Zisserman, "Three things everyone should know to improve object retrieval," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 2911–2918.

[74] P. F. Alcantarilla and T. Solutions, "Fast explicit diffusion for accelerated features in nonlinear scale spaces," *IEEE Trans. Patt. Anal. Mach. Intell*, vol. 34, no. 7, pp. 1281–1298, 2011.

[75] J. Wang, J. Fang, X. Liu, D. Zhao, and Q. Xiao, "A fast mosaic method for airborne images: the new template-convolution speed-up robust features (tsurf) algorithm," *International Journal of Remote Sensing*, vol. 35, no. 16, pp. 5959–5970, 2014.

[76] NumPy Developers, "numpy.linalg.norm," last accessed 13 May 2022. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.linalg.norm.html

[77] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[78] P. K. Chilukuri, P. Padala, P. Padala, V. S. Desanamukula, and P. R. Pvgd, "I, r-stitch unit: Encoder-decoder-cnn based image-mosaicing mechanism for stitching non-homogeneous image sequences," *IEEE Access*, vol. 9, pp. 16 761–16 782, 2021.

[79] L. Kang, Y. Wei, J. Jiang, and Y. Xie, "Robust cylindrical panorama stitching for low-texture scenes based on image alignment using deep learning and iterative optimization," *Sensors*, vol. 19, no. 23, p. 5310, 2019.

[80] W. Li, B. Hu, C. Song, F. Zhao, H. Ma, and Y. Wang, "An image stitching method for blades of wind turbine based on background removal preprocessing," in *2020 5th International Conference on Communication, Image and Signal Processing (CCISP)*. IEEE, 2020, pp. 174–178.

[81] B. LeBlanc, C. Niezrecki, P. Avitabile, J. Sherwood, and J. Chen, “Surface stitching of a wind turbine blade using digital image correlation,” in *Topics in Modal Analysis II, Volume 6*. Springer, 2012, pp. 277–284.

[82] B. LeBlanc, C. Niezrecki, P. Avitabile, J. Chen, J. Sherwood, and S. Hughes, “Full-field inspection of a wind turbine blade using three-dimensional digital image correlation,” in *Industrial and Commercial Applications of Smart Structures Technologies 2011*, vol. 7979. International Society for Optics and Photonics, 2011, p. 79790L.

[83] A. Fernández, *Mastering OpenCV 4 with Python*. Packt Publishing Ltd, 2019.

[84] Aryan Verma, “CLAHE Histogram Equalization - OpenCV,” last accessed 16 April 2022. [Online]. Available: http://geeksforgeeks.org/clahe-histogram-eqalization-opencv

[85] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes: Active contour models,” *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[86] V. Caselles, R. Kimmel, and G. Sapiro, “Geodesic active contours,” *International Journal of Computer Vision*, vol. 22, no. 1, pp. 61–79, 1997.

[87] M. Stokkeland, “A computer vision approach for autonomous wind turbine inspection using a multicopter,” Master’s thesis, Institutt for teknisk kybernetikk, 2014.

[88] Teledyne Princeton Instruments, “Field of View and Angular Field of View,” last accessed 13 May 2022. [Online]. Available: https://www.princetoninstruments.com/learn/camera-fundamentals/field-of-view-and-angular-field-of-view