Universitat
de les Illes Balears

DOCTORAL THESIS
2017

CONTRIBUTIONS TO ROBOT-BASED VESSEL VISUAL INSPECTION

Francisco Bonnín Pascual

Universitat
de les Illes Balears

DOCTORAL THESIS
2017

Doctoral Programme of Computer Science

CONTRIBUTIONS TO ROBOT-BASED VESSEL VISUAL INSPECTION

Francisco Bonnín Pascual

Thesis Supervisor: Dr. Alberto Ortiz Rodríguez

Doctor by the Universitat de les Illes Balears

Statement of Authorship

This thesis has been submitted to the *Escola de Doctorat*, *Universitat de les Illes Balears*, in fulfilment of the requirements for the degree of *Doctor en Informática*. I hereby declare that, except where specific reference is made to the work of others, the content of this dissertation is entirely my own work, describes my own research and has not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university.

Francisco Bonnín Pascual

Palma de Mallorca, June, 2017

Funding

The work reported in this thesis was supported by the European Social Fund through grant FPI10-43175042V (Conselleria d'Educació, Cultura i Universitats, Govern de les Illes Balears) and by FP7 projects MINOAS (GA 233715) and INCASS (GA 605200).

Supervisor's Agreement

I, Alberto Ortiz, Ph.D. in Computer Science and Associate Professor at the *Department of Mathematics and Computer Science, Universitat de les Illes Balears*

ATTEST THAT:

this dissertation, titled *Contributions to Robot-based Vessel Visual Inspection* and submitted by Francisco Bonnín Pascual for obtaining the degree of *Doctor en Informática*, was carried out under my supervision and contains enough contributions to be considered as a doctoral thesis.

Dr. Alberto Ortiz Rodríguez

Palma de Mallorca, June, 2017

Abstract

Vessels are nowadays one of the most cost effective ways to transport goods around the world. Despite the efforts to avoid maritime accidents, these still occur and, from time to time, have catastrophic consequences in environmental, human and/or economic terms. Structural failures caused by cracks and/or corrosion are the main cause of these accidents and, as such, vessels are submitted to periodical inspections in order to ensure their structural integrity. To carry out this task, vessels have to be emptied and situated in a dockyard where high scaffoldings are installed to allow the human inspectors to reach the highest parts of the vessel structure. Besides, the surveys are on many occasions performed in hazardous environments with difficult access. In economic terms, total expenses can reach up to one million dollars. Therefore, it is clear that any level of automation of the inspection process that can lead to a reduction of the inspection time, a reduction of the financial costs and/or an increase in the safety of the operation is fully justified. In this regard, this dissertation presents novel technological tools to contribute to re-engineering the process of vessel visual inspection. On the one hand, a novel aerial robotic platform is proposed to allow the surveyor to perform a proper inspection from a safe and comfortable position. It consists in an easy-to-use device which has been developed around the Supervised Autonomy paradigm, so that the vehicle is in charge of all the safety-related issues, while the surveyor provides the displacement commands and focuses on the inspection process. On the other hand, novel vision-based algorithms for defect detection on vessels structures are proposed. Firstly, several corrosion detection methods are described, based on the combination of different colour and texture descriptors. Secondly, a crack detection method, which combines edge detection with a region growing procedure, is proposed. Finally, the idea of saliency for detecting generic defects on vessel structures is evaluated, and also to improve the performance of the corrosion and crack detectors. The aerial platform and the defect detectors are evaluated both under laboratory conditions and during field tests performed on board real vessels. The results obtained allow to confirm the usability and the good performance of all the proposed technological tools.

Resumen

El transporte marítimo es una de las maneras más efectivas de transportar mercancías de un lugar a otro del mundo. Aunque hoy en día se llevan a cabo muchos esfuerzos para evitar los accidentes marítimos, estos todavía ocurren y, de vez en cuando, tienen consecuencias catastróficas en términos ambientales, humanos y/o económicos. Los daños estructurales causados por grietas y/o corrosión son la causa principal de estos accidentes y, por ello, los barcos son sometidos a inspecciones periódicas con el objetivo de garantizar su integridad estructural. Para llevar a cabo una inspección, los barcos son vaciados y llevados a un astillero donde se instalan andamiajes para permitir a los inspectores alcanzar las zonas más altas de su estructura. Estas inspecciones se realizan muchas veces en entornos peligrosos o de difícil acceso. En términos económicos, el proceso puede suponer un desembolso de hasta un millón de dolares. Por todo ello, cualquier contribución que suponga una reducción en el tiempo/coste de la inspección, o un incremento en la seguridad de los operarios, está justificada. En esta tesis se proponen nuevas herramientas tecnológicas que pretenden contribuir al rediseño de los procesos de inspección visual de barcos. Por un lado, se propone una nueva plataforma robótica aérea que permite al operario realizar la inspección del barco desde una posición segura y cómoda. Esta plataforma consiste en un dispositivo de fácil manejo que ha sido desarrollado siguiendo el paradigma de la Autonomía Supervisada, de tal manera que el vehículo se encarga de todas las tareas referentes a la seguridad, mientras que el operario proporciona las consignas de desplazamiento y puede centrarse en el proceso de inspección. Por otro lado, se proponen diversos algoritmos basados en visión para la detección de defectos en la estructura del barco. En primer lugar, se proponen varios métodos para la detección de corrosión, basados en la combinación de diferentes descriptores de color y de textura. En segundo lugar, se propone un algoritmo para la detección de grietas que combina la extracción de contornos con un proceso de crecimiento de regiones. Finalmente, se evalúa el concepto de notoriedad para la detección de defectos genéricos, y para la mejora del rendimiento de los detectores de corrosión y de grietas. La plataforma robótica y los detectores de defectos propuestos han sido evaluados tanto en laboratorio como durante pruebas de campo realizadas a bordo de un barco real. Los resultados obtenidos permiten confirmar la utilidad y el buen rendimiento de las diferentes herramientas tecnológicas propuestas.

Resum

El transport marítim és una de les maneres més efectives de transportar béns d'un lloc a l'altre del món. Encara que avui dia es realitzen grans esforços per tal d'evitar els accidents marítims, aquests encara ocorren i, de tant en tant, tenen conseqüències catastròfiques en termes ambientals, humans i/o econòmics. Els problemes estructurals causats per esquerdes i/o corrosió són la causa principal d'aquests accidents i, per això, els vaixells són sotmesos a ins-peccions periòdiques amb l'objectiu de garantir la seva integritat estructural. Per dur a terme una inspecció, els vaixells són buidats i portats a una drassana on s'instal·len bastides que permeten als inspectors arribar a les zones més altes de la seva estructura. Aquestes inspeccions es realitzen sovint en compartiments perillosos o de difícil accés. En termes econòmics, el procés pot suposar un desemborsament de fins a un milió de dòlars. Per tot això, qualsevol contribució que suposi una reducció en el temps/cost de la inspecció, o un increment en la seguretat dels operaris, està justificada. En aquesta tesi es proposen noves eines tecnològiques que pretenen contribuir al redisseny dels processos d'inspecció visual de vaixells. D'una banda, es proposa una nova plataforma robòtica aèria que permet a l'operari realitzar la ins-pecció del vaixell des d'una posició segura i còmoda. Aquesta plataforma consisteix en un dispositiu de fàcil ús que ha estat desenvolupat seguint el paradigma de l'Autonomia Supervisada, de tal manera que el vehicle s'encarrega de totes les tasques referents a la seguretat, mentre que l'operari proporciona les consignes de desplaçament i pot centrar-se en el procés d'inspecció. D'altra banda, es proposen diversos algorismes basats en visió per a la detecció de defectes en l'estructura del vaixell. En primer lloc, es proposen diversos mètodes per a la detecció de corrosió, basats en la combinació de diferents descriptors de color i de textura. En segon lloc, es proposa un algorisme per a la detecció d'esquerdes que combina l'extracció de contorns amb un procés de creixement de regions. Finalment, s'avalua el concepte de notorietat per a la detecció de defectes genèrics, i per a la millora del rendiment dels detectors de corrosió i d'esquerdes. La plataforma robòtica i els detectors de defectes proposats han estat avaluats tant en laboratori com durant proves de camp realitzades a bord d'un vaixell real. Els resultats obtinguts permeten confirmar la utilitat i el bon rendiment de les diferents eines tecnològiques propostes.

A n'Aina i na Neus, s'alegria des meu cor.

Acknowledgements

Let me write some words in Catalan. Voldria aprofitar aquestes línies per agrair a totes les persones que, d'una manera o d'una altra, m'han ajudat en la realització d'aquesta tesi. De manera especial, vull agrair:

- A n'Alberto Ortiz, el meu director de tesi, per tota l'ajuda i guiatge que m'ha donat durant aquests anys, per la seva capacitat d'escoltar, i per l'oportunitat que em dóna de treballar en el que a mi més m'agrada.
- A n'Emilio García, el meu company de batalla i de despatx, pels incomptables bons moments que hem passat junts durant aquests anys, pels pocs mals moments que també hem compartit, per tot el seu ajut, i per deixar-me fer servir la seva plantilla per la tesi.
- A nen Joan Pep Company, pel seu bon rotllo, per la seva aportació en el disseny i la mecanització de molts dels elements de la plataforma robòtica, i per la seva predisposició a donar un cop de mà en qualsevol moment.
- A nen Miquel, en Pep Lluís, en Francesc, n'Alberto, na Inés, n'Eric, en Toni i en David, per la seva companyonia, pels seus consells, i per tots els moments de desconnexió que hem compartit fent un cafè.
- A nen Javi Antich, per la seva capacitat d'alegrar-me el dia només amb unes paraules, pels seus ànims, i per fer-me guanyar algun partit de bàdminton de tant en tant.
- A la resta de membres del SRV, a n'Óscar Valero i a en Toni Mesquida, pel seu ajut, pel seu interés, i per l'atmosfera sana que ajuden a crear.
- A la meva família, especialment als meus pares, Pep i Maria Dolores, i als meus germans, Josep i Jaume, per la seva preocupació, pels seus ànims, per creure sempre en mi, i per fer possible que jo hagi arribat fins aquí.
- Finalment, vull agrair a la meva dona, n'Antònia, per tot el seu suport, per la seva paciència, i per la sobrecàrrega familiar a la qual l'he sotmesa durant aquests anys. Esper poder compensar tot el temps i l'esforç que has dedicat perquè jo pogués fer aquesta tesi.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

AUC Area Under the Curve

AUV Autonomous Underwater Vehicle

BoW Bag-of-Words

CAD Computer-Aided Design

CART Classification and Regression Trees

CMOS Complementary Metal-Oxide-Semiconductor

CNN Convolutional Neural Network

DIDSON Dual-Frequency Identification Sonar

DOF Degrees Of Freedom

DVL Doppler Velocity Log

dwt Deadweight Tonnage

DWT Discrete Wavelet Transform

EKF Extended Kalman Filter

ELM Extreme Learning Machine

ESEIF Exactly Sparse Extended Information Filter

FFT Fast-Fourier Transform

FHT Fast Haar Transform

FLDA Fisher Linear Discriminant Analysis

FN False Negative

FP False Positive

FPR False Positive Rate

FPSO Floating Production, Storage and Offloading

GDP Gross Domestic Product

GLC Generic Linear Constraints

GNSS Global Navigation Satellite System

GPS Global Positioning System

GT Ground Truth

HLP High-Level Processor

HOG Histogram of Oriented Gradients

HSI Hue-Saturation-Intensity

HSV Hue-Saturation-Value

IBVS Image-Based Visual Servoing

ICP Iterative Closest Point

IMU Inertial Measurement Unit

INCASS INspection CApabilities for enhanced Ship Safety

IR Infrared

KF Kalman Filter

LBL Long-BaseLine

LED Light-Emitting Diode

LiDAR Light Detection And Ranging

LLP Low-Level Processor

LMS Least Mean Square

LoG Laplacian of Gaussian

LOOCV Leave-One-Out Cross-Validation

MAV Micro-Aerial Vehicle

MBD Model Based Design

MINOAS Marine INspection rObotic Assistant System

NDT Non-Destructive Testing

NN Neural Network

OECD Organization for Economic Cooperation and Development

PBVS Position-Based Visual Servoing

PCA Principal Component Analysis

PDF Probability Density Function

PID Proportional-Integral-Derivative

PR Precision-Recall

PSNR Peak Signal-to-Noise Ratio

PTZ Pan-Tilt-Zoom

RGB Red-Green-Blue

ROC Receiver Operating Characteristic

ROI Region Of Interest

ROS Robot Operating System

ROV Remotely Operated Vehicle

SA Supervised Autonomy

SfM Structure from Motion

SLAM Simultaneous Localization and Mapping

SRV Systems, Robotics and Vision group

SS1/2/3 Sensor Suite 1/2/3

SUAS Small Unmanned Aerial Systems

SUN Saliency Using Natural Statistics

SVM Support Vector Machine

TN True Negative

TP True Positive

TPR True Positive Rate

UAV Unmanned Aerial Vehicle

UIB University of the Balearic Islands

US Ultrasound

UWB Ultra-Wide Band

VLCC Very Large Crude Carrier

VTOL Vertical Take-Off and Landing

WDFT Windowed Discrete Fourier Transform

Symbols and Notation

XYZ	Generic coordinate frame axes. For the case of a robot body frame, respectively, longitudinal, lateral and vertical axes.
$x/y/z$	Position along the $X/Y/Z$ axis in a specific coordinate frame
$\varphi/\theta/\psi$	Roll/pitch/yaw. Rotation around the $X/Y/Z$ axis of a body coordinate frame
$\dot{x}/\dot{y}/\dot{z}$	Linear velocity along the $X/Y/Z$ axis of a body coordinate frame
$\dot{\varphi}/\dot{\theta}/\dot{\psi}$	Angular velocity around the $X/Y/Z$ axis of a body coordinate frame
$\ddot{x}/\ddot{y}/\ddot{z}$	Linear acceleration along the $X/Y/Z$ axis of a body coordinate frame
$b_{\ddot{x}}/b_{\ddot{y}}/b_{\ddot{z}}$	Acceleration bias in the $X/Y/Z$ axis of a body coordinate frame
$d_b/d_f/d_l/d_r$	Distance to an obstacle below/in front of/to the left of/to the right of the robot
d	Distance measure
Δd	Difference between two consecutive distance measures
$\mathcal{T}$	Thrust
$\mathcal{T}_h$	Thrust to achieve hovering
$\Delta\mathcal{T}_h$	Hovering thrust update
$\mathcal{T}_{h\_incr}$	Maximum hovering thrust update allowed
$\varphi_d/\theta_d/\dot{\psi}_d/\mathcal{T}_d$	Desired roll/pitch/yaw velocity/thrust command
$\dot{x}_d/\dot{y}_d/\dot{z}_d$	Desired linear velocity along the $X/Y/Z$ axis
$\dot{x}_{d_M}/\dot{y}_{d_M}/\dot{z}_{d_M}$	Maximum desired velocity allowed in the $X/Y/Z$ axis
$\dot{x}_{ud}/\dot{y}_{ud}/\dot{z}_{ud}$	Desired linear velocity in the $X/Y/Z$ axis defined by the user
$\dot{\psi}_{ud}$	Desired angular velocity around the Z axis defined by the user
$\dot{x}_{d\_ag}/\dot{x}_{d\_ai}$	Desired velocity in the X axis provided by the *attenuated_go/inspect* behaviour
$\dot{x}_{d\_pc}$	Desired velocity in the X axis provided by the *prevent_collision* behaviour

$\dot{z}_{d\_lmh}$	Desired velocity in the Z axis provided by the *limit_max_height* behaviour
$\dot{x}_{d\_ers}/\dot{z}_{d\_ers}$	Desired velocity in the X/Z axis provided by the *ensure_reference_surface_detection* behaviour
$\dot{\psi}_{d\_ers}$	Desired angular velocity around the Z axis provided by the *ensure_reference_surface_detection* behaviour
K_{ag}	Attenuation factor used in the *attenuated_go* behaviour
K_{pc}	Repulsion factor used in the *prevent_collision* behaviour
K_{lmh}	Attraction factor used in the *limit_max_height* behaviour
K_{ers}	Attraction factor used in the *ensure_reference_surface_detection* behaviour
d_m	Minimum distance to obstacles allowed
z_M	Maximum flight height allowed
d_{b_M}/d_{f_M}	Maximum distance to the ground/front wall allowed
v_m	Minimum battery voltage for flying
$\mathcal{E}$	Error
$\mathcal{E}_{\dot{x}}/\mathcal{E}_{\dot{z}}/\mathcal{E}_{\dot{y}}$	Error in the linear velocity along the $X/Y/Z$ axis
$\mathcal{E}_z$	Error in the position along the Z axis
t	Time
Δt	Time between two consecutive samples
u	Control signal. Output of a controller
$K_p/K_d/K_i$	Coefficient for the proportional/derivative/integral term in a PID controller
$K_p^{\dot{x}}/K_d^{\dot{x}}/K_i^{\dot{x}}$	Coefficient for the proportional/derivative/integral term in PID controller in charge of controlling the linear velocity along the X axis
$K_p^{\dot{y}}/K_d^{\dot{y}}/K_i^{\dot{y}}$	Coefficient for the proportional/derivative/integral term in PID controller in charge of controlling the linear velocity along the Y axis
$K_p^{\dot{z}}/K_d^{\dot{z}}/K_i^{\dot{z}}$	Coefficient for the proportional/derivative/integral term in PID controller in charge of controlling the linear velocity along the Z axis
$K_p^z/K_d^z/K_i^z$	Coefficient for the proportional/derivative/integral term in PID controller in charge of controlling the position along the Z axis
k	Discrete time instant
$\mathbf{x}_k$	State at time k
$\mathbf{F}_k$	State transition model at time k within a Kalman Filter
$\mathbf{w}_k$	Process noise at time k within a Kalman Filter

$\mathbf{Q}_k$	Process noise covariance at time k within a Kalman Filter
$\mathbf{z}_k$	Sensor measurement at time k
$\mathbf{H}_k$	Observation model at time k within a Kalman Filter
$\mathbf{v}_k$	Observation noise at time k within a Kalman Filter
$\mathbf{R}_k$	Observation noise covariance at time k within a Kalman Filter
σ	Covariance
$\sigma_{dist}/\sigma_{vel}$	Covariance of a distance/velocity measure
λ	Scale factor
α	Angle
α_{incr}	Increment between two steps in an angular range
$\alpha_{min}/\alpha_{max}$	Minimum/maximum value of an angular range
p/q	Point/pixel in a space/image
b	Beam in a laser scan
$\mathcal{P}$	Point cloud
$\mathcal{T}$	Rigid transform, including a translation and a rotation
μ^+/μ^-	Mean of positive/negative values
D_B/BC	Bhattacharyya distance/coefficient
$\tau_C/\tau_D/\tau_E$	Colour/dissimilarity/energy threshold within a corrosion detector
τ_G/τ_L	Gray-level/elongation threshold within a crack detector
γ	Proportional factor used to define the window size within a crack detector
$I/R/G/B$	Intensity/red/green/blue pre-feature map
$\widehat{I}/\widehat{R}/\widehat{G}/\widehat{B}$	Pyramid generated from $I/R/G/B$ pre-feature map
$\widehat{O}_{0/45/90/135}$	Pyramid generated from I using an oriented Gabor filter with orientation $0/45/90/135°$.
$\boldsymbol{I}/\boldsymbol{C}/\boldsymbol{O}$	Feature map for contrast in intensity/colour/orientation
$\overline{\boldsymbol{I}}/\overline{\boldsymbol{C}}/\overline{\boldsymbol{O}}$	Normalized feature map for contrast in intensity/colour/orientation
$\boldsymbol{D}_{con}/\boldsymbol{D}_{sym}$	Defect map based on contrast/symmetry
$\boldsymbol{D}_{AND}$	Defect map obtained using the AND operator
$\boldsymbol{D}_{OR}$	Defect map obtained using the OR operator
$\boldsymbol{D}_{ORA}$	Defect map obtained using the ORA operator

S_z	Saliency at image point z
f_z	Feature values at image point z
τ_S	Saliency threshold

Chapter 1

Introduction

1.1 Scope of Research

The Systems, Robotics and Vision group (SRV) of the University of the Balearic Islands (UIB) has been involved in the European projects *MINOAS* (Marine INspection rObotic Assistant System), from 2009 to 2012, and *INCASS* (INspection CApabilities for enhanced Ship Safety), from 2013 to 2017. The main goal of both projects is to provide new tools and methodologies to improve and facilitate the inspection of vessels.

1.2 Vessels and Maritime Transport

Vessels constitute one of the most cost effective forms of transporting goods around the world. The seaborne trade increases year after year pushed by the global economic growth [1]. By way of example, Fig. 1.1 shows in a timeline the relation between the seaborne shipments, the merchandise trade, the industrial activity, as measured by the Organization for Economic Cooperation and Development (OECD) Industrial Production Index, and the world economic growth, indicated in terms of Gross Domestic Product (GDP). As can be seen, the four variables share the same tendency.

The increase of seaborne trade in the last decades is detailed in Fig. 1.2, which provides the millions of tons loaded per year differentiating the four main types of cargo: bulks, oil and gas, containers and other dry cargo. The amount of shipments of these four main cargo types determine the structure of the international seaborne trade.

Each cargo category requires from an specific type of vessel, resulting in four different principal vessel types: bulk carriers (for iron ore, coal, grain and other minor bulks), tankers (for crude oil, petroleum products, gas and chemical products), container ships and general cargo. These four vessel types constitute around 90% of the world fleet deadweight tonnage (dwt). This measure indicates how much mass a ship can safely carry, excluding the weight of the ship. In this regard, Fig. 1.3 provides the evolution of the percentage share of dwt for these vessel types in the last decades.

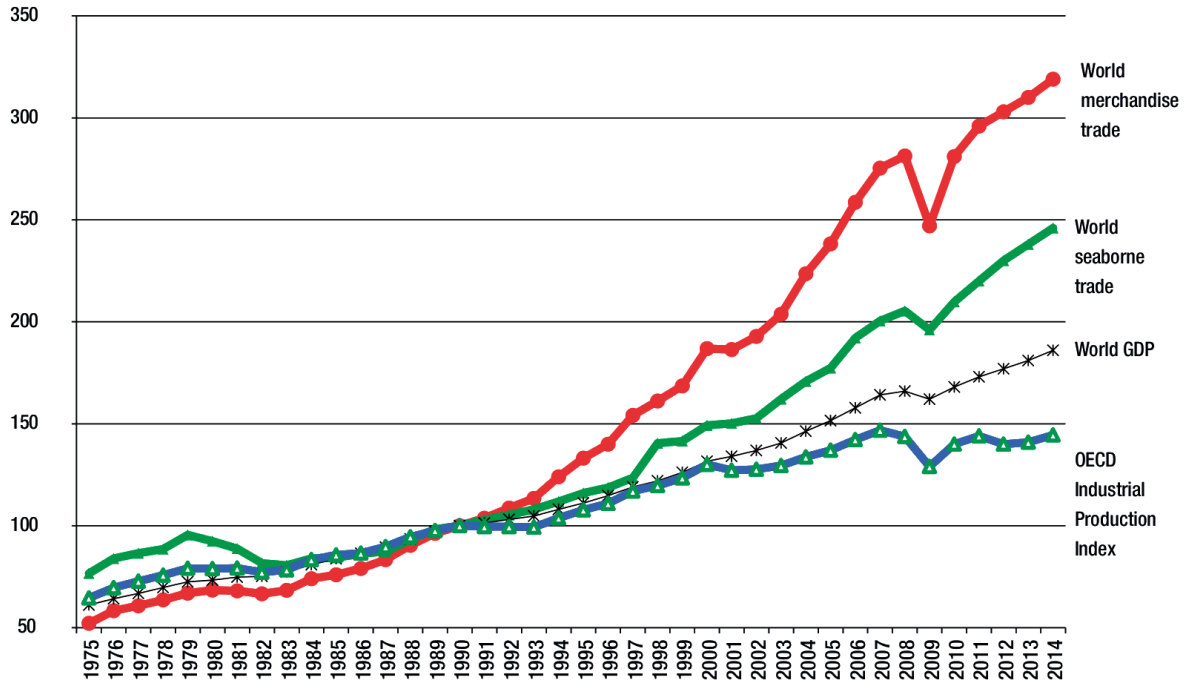


Figure 1.1: The OECD Industrial Production Index and indices for world GDP, merchandise trade and seaborne shipments (1975-2014) (base year 1990 = 100). Figure taken from [1].

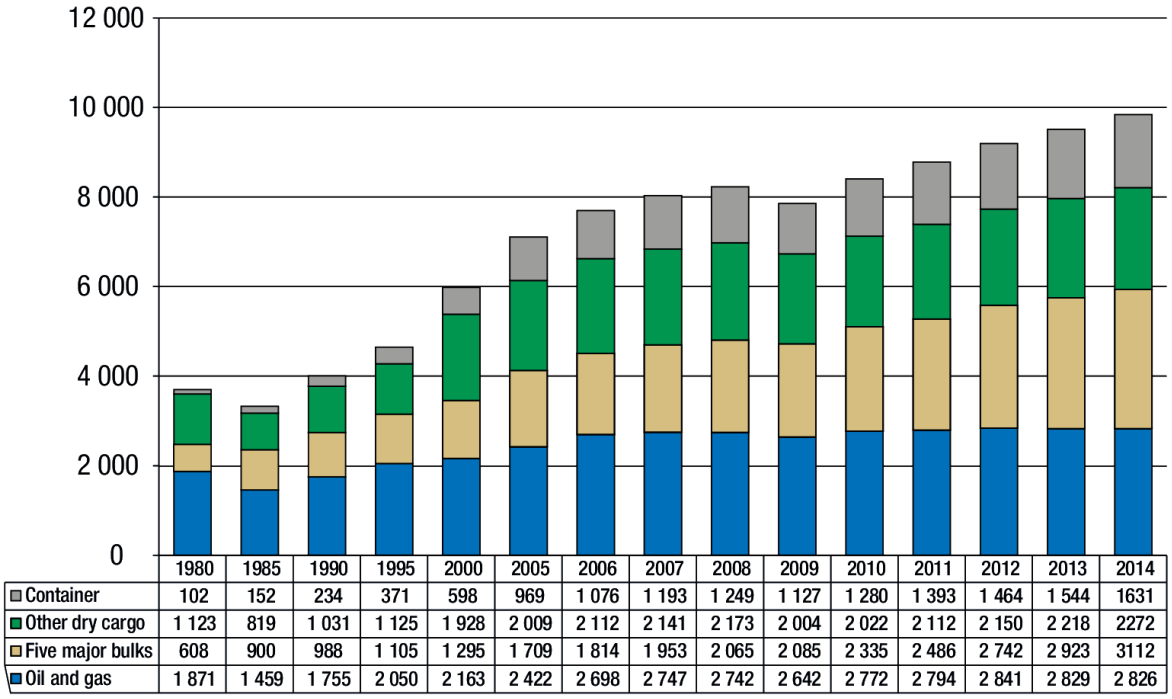
1.2.1 Defects on Vessel Structures

A vessel hull can be affected by different kinds of defects that may appear due to several factors [2]. The following categories are used to group vessel hull defects according to its cause:

- *Structural overload*, which comprises defects caused by placing greater stress on the ship than it was designed for. Defects on this category may be the result of grounding, collision, contact (with the quay, with tugs, etc.), operational overload (for example, poor loading sequence, too high a rate of loading, variable ballast levels during loading) or overload due to heavy weather.

- *Design related defects*, which can result from differences between the actual loads experienced by the structure compared with the theoretical loads used for design. Defects in this category also appear when design tolerances are exceeded, applicable standards are not compiled or due to the existence of inadequacies in the initial design. This category of defect is characterised by the defect having no apparent cause and by it repeatedly re-occurring following repair.

- *Workmanship related defects*, which are caused by the use of sub-standard materials, poor alignment, poor welding, poor finishing and/or omissions, and initial deformations.



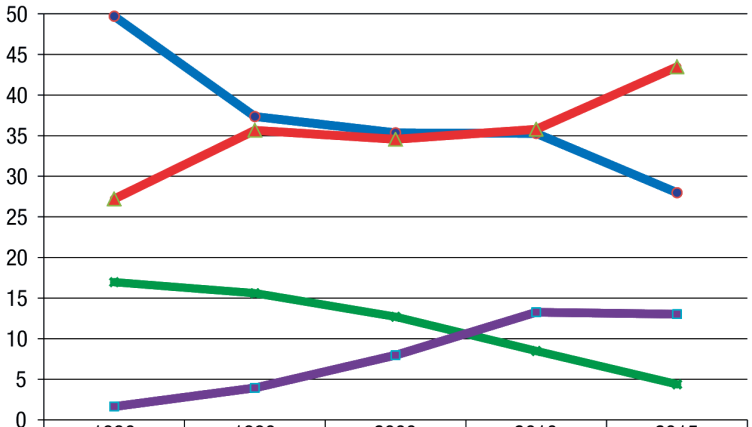
	1980	1985	1990	1995	2000	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014
Container	102	152	234	371	598	969	1 076	1 193	1 249	1 127	1 280	1 393	1 464	1 544	1631
Other dry cargo	1 123	819	1 031	1 125	1 928	2 009	2 112	2 141	2 173	2 004	2 022	2 112	2 150	2 218	2272
Five major bulks	608	900	988	1 105	1 295	1 709	1 814	1 953	2 065	2 085	2 335	2 486	2 742	2 923	3112
Oil and gas	1 871	1 459	1 755	2 050	2 163	2 422	2 698	2 747	2 742	2 642	2 772	2 794	2 841	2 829	2 826

Figure 1.2: International seaborne trade in selected years. Values in millions of tons loaded. Figure taken from [1].

Defects in this category appear regardless of the quality of the design.

- *Vibration related*, comprising fatigue defects resulting from hydrodynamic or mechanically induced vibration.
- *Corrosion*, including general wastage and localised corrosion. Local coating breakdowns are more likely in areas which are hard to access and therefore maintain. Corrosion rates are accelerated in areas of higher stresses (corrosion under stress). In turn, the more a structure corrodes, the greater are the stresses on the remaining undamaged structure and therefore the corrosion rate increases. Conversely, if the stresses in an area are reduced, the rate of corrosion is also reduced.
- *Pitting*, which are defects generally caused by corrosion. This defect, indeed, is typically considered as a kind of corrosion. It is further explained later in this section.

From a more simpler point of view, and regardless of its cause, two main defective situations can be considered: cracks and corrosion. On the one hand, cracks generally develop at intersections of structural items or discontinuities (including changes in thickness) due to stress concentration, although they also may be related to material or welding defects. If the crack remains undetected and unrepaired, it can grow to a size where it can cause sudden



	1980	1990	2000	2010	2015
Oil tanker	49.7	37.4	35.4	35.3	28.0
Dry bulk	27.2	35.6	34.6	35.8	43.5
General cargo	17.0	15.6	12.7	8.5	4.4
Container	1.6	3.9	8.0	13.3	13.0

Figure 1.3: World fleet percentage share of dwt by principal vessel types, 1980-2015 (beginning-of-year figures). Note: All propelled seagoing merchant vessels of 100 tons and above, excluding inland waterway vessels, fishing vessels, military vessels, yachts and offshore fixed and mobile platforms and barges (with the exception of floating production, storage and offloading units [FPSO] and drillships). Figure taken from [1].

fracture. Therefore, care is needed to visually discover fissure occurrences in areas prone to high stress concentration. Specific area of occurrence of cracks includes:

- hard points (see Fig. 1.4),
- ends of brackets and stiffeners (see Fig. 1.5 [yellow] and Fig. 1.6 [A]),
- change of section (see Fig. 1.6 [B]),
- change of thickness (see Fig. 1.6 [C]),
- openings (see Fig. 1.6 [D]),
- misalignments,
- three planes (see Fig. 1.5 [red]) and
- weld defect in the structure.

On the other hand, different kinds of corrosion may arise in vessel structures:

- *general corrosion*, that appears as non-protective friable rust which can occur uniformly on uncoated surfaces;

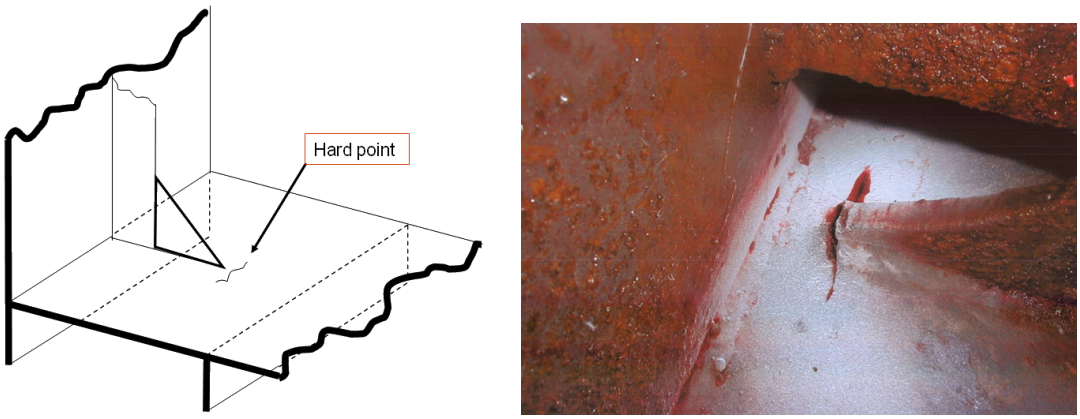


Figure 1.4: Detail of a crack at a hard point.

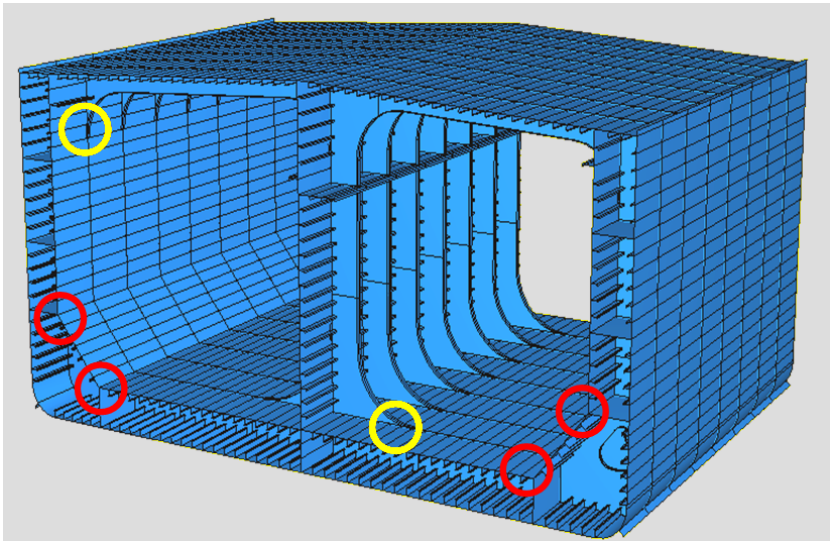


Figure 1.5: Specific areas of occurrence of cracks within a tank: (yellow) end of brackets and stiffeners and (red) three planes.

- *pitting*, a localized process that is normally initiated due to local breakdown of coating and that derives, through corrosive attack, in deep and relatively small diameter pits that can in turn lead to hull penetration in isolated random places (see Fig. 1.7);

- *grooving*, again a localized process, but this time characterized by linear shaped corrosion which occurs at structural intersections where water collects and flows; and

- *weld metal corrosion*, which affects the weld deposits, mostly due to galvanic action with the base metal, and are likelier in manual welds than in machine welds.

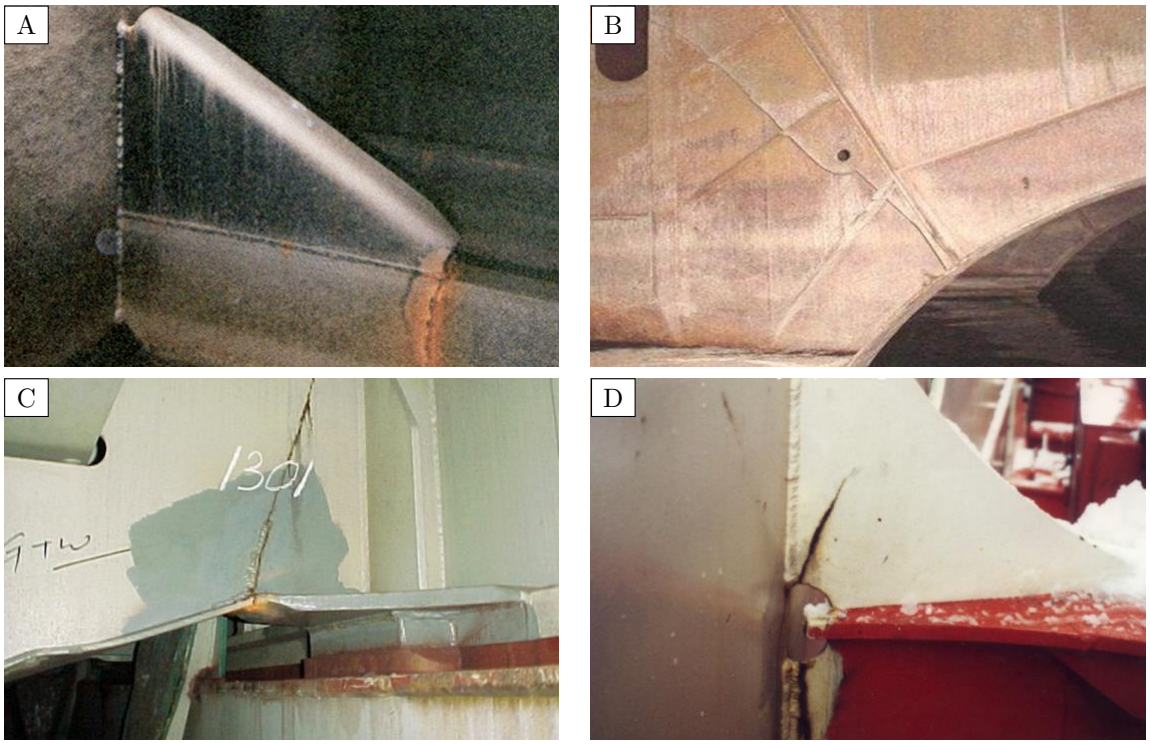


Figure 1.6: Specific areas of occurrence of cracks: (A) end of bracket, (B) change of section, (C) change of thickness and (D) opening.

Figure 1.7: Examples of pitting corrosion.

1.2.2 Vessel Structure Maintenance

An early detection of defects on vessel structures prevent these from buckling and/or fracturing. For this reason, extensive inspection schemes are carried out to assess the structural integrity of vessels. Ship hull inspections are currently conducted either as part of *Class surveys* (where Class refers to Classification Societies, also known as Shipping Registers) or *Condition surveys* [2]. Surveys under the first category are conducted by Class experts and

are described by a thorough set of rules. The general purpose of the survey is to monitor the deterioration of the structure and describe maintenance activities as part of a preventive strategy in order to satisfy sea-worthiness criteria. Condition surveys are similarly focused on the determination of the preventive actions from the owner's/operator's side to satisfy the requirements that retain the ship operational. Since the second group is largely based on the surveyor's experience and usually deals with focused activities, it does not have a formal structure. It does though roughly follow the list of activities determined for the corresponding Class survey.

Hull condition surveys basically rely on the interpretation of visual data, supported by the quantitative information of thickness measurements. The interpretation of visual information relies heavily on the experience and adeptness of the surveyor, while an indicative set of images is collected to support the findings. Depending on the type of survey (ship age and overall status) thickness data are collected and documented in appropriate forms, accompanying the results of the inspection. Condition surveys do not follow the same formalism, but rely on the same techniques to identify areas of interest. The equipment used during the conduction of surveys essentially consists of:

- an Ultrasonic Thickness Measurement (UTM) device,
- a camera used for grabbing some indicative images and
- a tablet-computer equipped with specialized software to support the survey activities (list of regulations, previous corresponding reports, etc).

Additionally, to determine the state of a corroded structure, it is common to estimate the corrosion level as percentage of affected area. Traditional methods quantify corrosion by visual comparison of the area under study with various dot patterned charts, depending on corrosion type (localized, scattered or linear). An example of a schematic guide for the evaluation of the pitting intensity is presented in Fig. 1.8.

To perform a complete hull inspection, the vessel has to be emptied and situated in a dockyard (and probably in a dry-dock, see Fig. 1.9 [left]), where typically temporary staging, lifts, movable platforms, etc. need to be installed to allow the workers for close-up inspection of all the different metallic surfaces and structures (see Fig. 1.9 [right]). The items to survey depend on the type and age of vessel, as well as the kind of survey that is being carried out. To illustrate the enormity of the inspection task, the surveying of a central cargo tank on a Very Large Crude Carrier (VLCC), involves checking over 860 m of web frames (primary stiffening members) and approximately 3.2 Km of longitudinal stiffeners; and the complete survey to assess the state of the whole vessel can mean the visual assessment of more than 600.000 m^2 of steel.

Furthermore, the surveys are on many occasions performed in a potentially hazardous environment with both flammable and toxic gases and significant heights involved. As a result,

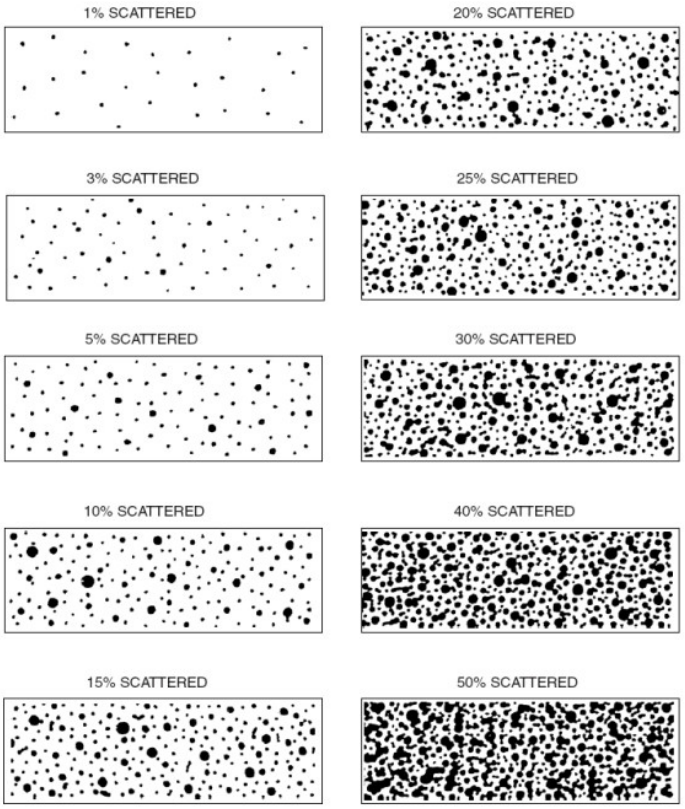


Figure 1.8: Pitting intensity diagrams. Figure taken from [2].

although accidents are extremely rare, when they do arise they can have serious consequences. Due to these complications, the total cost of a single surveying can exceed $1M once you factor in the vessel's preparation, use of yard's facilities, cleaning, ventilation, and provision of access arrangements. In addition, the owners experience significant lost opportunity costs while the ship is inoperable.

1.3 Facilitating the Visual Inspection of Vessels

As mentioned before, ship hull inspection activities have well established practices composed of tasks that do not include advanced technological tools. They mainly rely on human expertise and require the transfer of humans close to the structure in order to visually assess and/or take a measure. Considering the large metallic areas onboard vessels, providing the means of access requires a considerable effort, namely the use of scaffolding or cherry-pickers with corresponding safety issues for the human personnel. It is clear that the use of robotic platforms may help in that sense. By way of example, a robotic device equipped with cameras could be used to take pictures/videos of the vessel hull, so that the human surveyor does not need to physically reach the structures and surfaces for its visual inspection. Notice that,

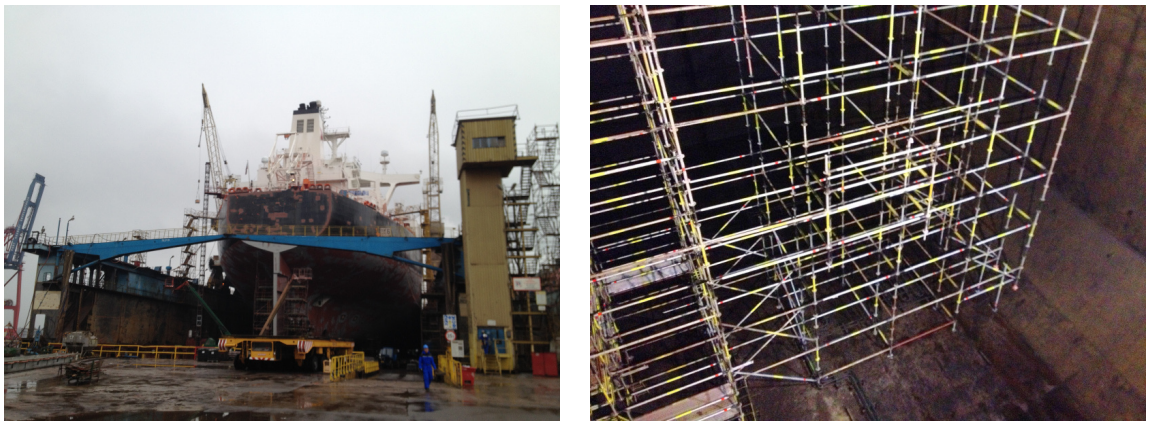


Figure 1.9: Tanker-type vessel prepared for a periodical survey: (left) vessel situated in a dry-dock, (right) scaffoldings installed inside a cargo hold.

besides scaffoldings and other temporary structures might result unnecessary, robotic assisted inspections might be carried out during sailing (as long as the inspected hold were empty), reducing the number of times that a vessel has to be driven to a shipyard for survey.

Moreover, image processing techniques could be of application to assist the surveyors during the assessment of the inspected surfaces. By way of example, images and videos captured using a robotic device may be processed in order to provide the surveyor an estimation of the state of the hull structure, indicating e.g. the appearance of a crack, or the percentage of corrosion detected.

In conclusion, it is clear that the use of advanced technological tools could offer an important reduction in economic and temporal terms, while increasing safety and comfort to the surveyors.

1.4 Objectives of the Thesis

The principle goal of this thesis is to obtain new advanced technological tools to contribute to the re-engineering process of vessel hulls inspection. Two main aspects are considered:

- The design and development of a robotic aerial platform to allow the surveyor to perform a proper visual inspection of the vessel hull from a safe and comfortable position. The idea is to provide a novel easy-to-use device that fulfils the requirements to operate inside cargo holds and tanks. The platform should obey the surveyor commands while it is in charge of all safety related issues, e.g. avoid collisions with ship structures or other obstacles. This entails the selection/design of the appropriate hardware and control software that fulfils all these requirements. Moreover, we wish to adapt a systematic approach and be as formal as possible from the very first moment, both at the hardware and software levels. Consequently this objective splits into the following subobjectives:

  - identify suitable robotic platforms,
  - identify adequate navigation sensor suites,
  - select an appropriate structure for the control software, and
  - formalize and design the required control software components.

- The design and implementation of novel vision-based algorithms devised to detect defects in color images taken from vessel structures. We focus on the detection of the two main defective situations, i.e. corrosion and cracks. As with the previous objective, we intend to adopt a systematic approach to achieve this goal. Therefore, the following subobjectives arise:
  - identify suitable features to detect corrosion in images,
  - devise and design corrosion detection methods following different approaches,
  - identify suitable features to detect cracks in images,
  - devise and design crack detection methods following different approaches,
  - identify possible features to detect generic defects in vessel structures, and
  - devise and design different approaches for generic defect detection, which are able to detect both corrosion and cracks in vessel structures.

1.5 Contributions

To fulfil the previous objectives, this dissertation presents the results of a research process which can be summarized in the following contributions:

- A complete survey of all the previous works related with the design or utilization of robotic platforms for the inspection of vessels. The reviewed approaches are classified depending on whether they are prepared for the underwater or above-water operation, the capabilities that they offer, the sensors used, etc.
- A review of existing approaches regarding the utilization of aerial robotic platforms for the visual inspection of infrastructures. These are classified depending on the kind of infrastructure inspected, the vehicle capabilities, the sensors employed and the output that they provide.
- A review of existing vision-based algorithms for the autonomous detection of corrosion or cracks in digital images. The methods are classified according to the technique which they are based on.
- A novel aerial robotic platform for vessel visual inspection. The control architecture is designed to allow the user/pilot to focus on the inspection at hand, while the robot

is in charge of all the safety related issues, such as collision avoidance or battery level monitoring. This framework also allows for an easy and comprehensive communication between the user and the platform, which results in a useful and easy-to-use tool.

- A collection of novel vision-based corrosion detection algorithms for the visual inspection of vessel metallic structures. These are based on the combination of different color and texture descriptors.

- Novel vision-based crack detection algorithms for the visual inspection of vessel metallic structures. It is based on the combination of edge detection and region-growing procedures.

- A collection of new saliency-based generic defect detectors intended for the inspection of vessel structures. Contrast and symmetry features are considered/combined within different frameworks.

- The improved versions of the corrosion and crack detectors previously presented, by means of boosting their performance through the combination of a generic saliency-based defect detector with each specific defect detector.

1.6 Document Overview

To present the preceding contributions, the dissertation is divided into six chapters as follows:

- **Chapter 2** reviews previous works related with the key aspects of our research, namely, robotic platforms used for inspection and vision-based algorithms devised for defect detection.

- **Chapter 3** extensively presents the requirements, design, implementation and performance evaluation of a novel aerial robotic device intended for vessel visual inspection.

- **Chapter 4** presents a collection of novel vision-based algorithms for defect detection on vessel structures. Three different research lines are considered. Firstly, we focus on specific techniques for corrosion detection; secondly, we deal with the detection of cracks; and, finally, we asses the idea of using visual saliency for detecting generic defects on vessel structures (i.e. considering both corrosion and cracks).

- **Chapter 5** evaluates the new vessel inspection tools during field tests performed on board a real vessel.

- **Chapter 6** concludes the dissertation by summarizing the main contributions and suggesting some future work to extend the research.

1.7 Related Publications

Parts of this thesis have been published in international journals and conference proceedings. The following list gives an overview about the individual publications.

Journal Articles

- Francisco Bonnin-Pascual and Alberto Ortiz, **A Novel Approach for Defect Detection on Vessel Structures using Saliency-related Features**, conditionally accepted at *Ocean Engineering*.
- Alberto Ortiz, Francisco Bonnin-Pascual, Emilio Garcia-Fidalgo and Joan P. Company-Corcoles, **Vision-Based Corrosion Detection Assisted by a Micro-Aerial Vehicle in a Vessel Inspection Application**, *Sensors*, vol. 16, no. 2118, 2016, ISSN 1424-8220 [3].
- Francisco Bonnin-Pascual and Alberto Ortiz, **A Flying Tool for Sensing Vessel Structure Defects using Image Contrast-based Saliency**, *IEEE Sensors Journal*, vol. 16, no. 15, pp. 6114-6121, 2016, ISSN 1530-437X [4].

Conference Proceedings and Workshops

- Francisco Bonnin-Pascual and Alberto Ortiz, **A Saliency-boosted Corrosion Detector for the Visual Inspection of Vessels**, *International Conference of the Catalan Association for Artificial Intelligence (CCIA)*, Deltebre, Tarragona (Spain), 2017 [5].
- Alberto Ortiz, Francisco Bonnin-Pascual, Emilio Garcia-Fidalgo and Joan P. Company-Corcoles, **Defect-level Inspection Aids for Automated Vessel Visual Inspection**, *Jornadas Automar (Marine Automation Workshop)*, Castellón, (Spain), 2017 [6].
- Alberto Ortiz, Francisco Bonnin-Pascual, Emilio Garcia-Fidalgo and Joan P. Company-Corcoles, **The INCASS Project Approach towards Automated Visual Inspection of Vessels**, *Jornadas Nacionales de Robótica* (Spanish Robotics Workshop), Valencia, (Spain), 2017 [7].
- Alberto Ortiz, Francisco Bonnin-Pascual, Emilio Garcia-Fidalgo and Joan P. Company-Corcoles, **Towards Automated Ship Inspection: A Visual Data-Oriented Toolbox**, *International Conference on Maritime Safety and Operations (MSO)*, Glasgow, Scotland (United Kingdom), 2016 [8].
- Francisco Bonnin-Pascual and Alberto Ortiz, **A Generic Framework for Defect Detection on Vessel Structures based on Image Saliency**, *IEEE International Conference on Emerging Technologies and Factory automation (ETFA)*, Berlin (Germany), 2016 [9].

- Thomas Koch, Sankaranarayanan Natarajan, Felix Bernhard, Alberto Ortiz, Francisco Bonnin-Pascual, Emilio Garcia-Fidalgo and Joan P. Company-Corcoles, **Advances in Automated Ship Structure Inspection**, *International Conference on Computer and IT Applications in the Maritime Industries (COMPIT)*, Lecce (Italy), 2016 [10].

- Alberto Ortiz, Francisco Bonnin-Pascual, Emilio Garcia-Fidalgo and Joan P. Company-Corcoles, **Visual Inspection of Vessels by means of a Micro-Aerial Vehicle: an Artificial Neural Network Approach for Corrosion Detection**, *Iberian Robotics Conference (ROBOT)*, Lisbon (Portugal), 2015 [11].

- Alberto Ortiz, Francisco Bonnin-Pascual, Emilio Garcia-Fidalgo and Joan P. Company-Corcoles, **Saliency-driven Visual Inspection of Vessels by means of a Multi-rotor**, Workshop on Vision-based Control and Navigation of Small, Lightweight UAVs, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg (Germany), 2015 [12].

- Francisco Bonnin-Pascual, Alberto Ortiz, Emilio Garcia-Fidalgo and Joan P. Company-Corcoles, **A Micro-Aerial Platform for Vessel Visual Inspection based on Supervised Autonomy**, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg (Germany), 2015 [13].

- Francisco Bonnin-Pascual and Alberto Ortiz, **A Probabilistic Approach for Defect Detection based on Saliency Mechanisms**, *IEEE International Conference on Emerging Technologies and Factory automation (ETFA)*, Barcelona (Spain), 2014 [14].

Technical Reports

- Francisco Bonnin-Pascual and Alberto Ortiz, **Detection of Defects on Vessel Structures using Saliency-related Features**, Department of Mathematics and Computer Science, University of the Balearic Islands, Tech. Rep. A-04-2015 [15].

- Francisco Bonnin-Pascual, Alberto Ortiz, Emilio Garcia-Fidalgo and Joan P. Company-Corcoles, **A Micro-Aerial Vehicle based on Supervised Autonomy for Vessel Visual Inspection**, Department of Mathematics and Computer Science, University of the Balearic Islands, Tech. Rep. A-02-2015 [16].

Other publications, reporting preliminary versions of some of the contributions presented in this dissertation, are listed below.

Journal Articles

- Markus Eich, Francisco Bonnin-Pascual, Emilio Garcia-Fidalgo, Alberto Ortiz, Gabriele Bruzzone, Yannis Koveos and Frank Kirchner, **A Robot Application for Marine**

Vessel Inspection, *Journal of Field Robotics*, vol. 31, no. 2, pp. 319-341, 2014, ISSN 1556-4959 [17].

- Alberto Ortiz, Francisco Bonnin-Pascual and Emilio Garcia-Fidalgo, **Vessel Inspection: A Micro-Aerial Vehicle-based Approach**, *Journal of Intelligent and Robotic Systems*, vol. 76, no 1, pp. 151-167, 2014, ISSN 0921-0296 [18].

Book Chapters

- Francisco Bonnin-Pascual and Alberto Ortiz, **Corrosion Detection for Automated Visual Inspection**, *Developments in Corrosion Protection*, Ed. InTech, pp. 619-632, 2014, ISBN 978-953-51-1223-5 [19].

Conference Proceedings and Workshops

- Francisco Bonnin-Pascual, Emilio Garcia-Fidalgo and Alberto Ortiz, **Semi-Autonomous Visual Inspection of Vessels Assisted by an Unmanned Micro Aerial Vehicle**, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Algarve (Portugal), 2012 [20].

- Emilio Garcia-Fidalgo, Francisco Bonnin-Pascual and Alberto Ortiz, **A Control Architecture for a Micro Aerial Vehicle Intended for Vessel Visual Inspection**, *Jornadas de Computación Empotrada (JCE)*, Elche (Spain), 2012 [21].

- Alberto Ortiz, Emilio Garcia-Fidalgo and Francisco Bonnin-Pascual, **A Micro Aerial Vehicle for Vessel Visual Inspection Assistance**, *International Conference on Computer and IT Applications in the Maritime Industries (COMPIT)*, Liege (Belgium), 2012 [22].

- Alberto Ortiz, Francisco Bonnin-Pascual and Emilio Garcia-Fidalgo, **On the Use of UAVs for Vessel Inspection Assistance**, *Workshop on Research, Development and Education on Unmanned Aerial Systems (RED-UAS)*, Sevilla (Spain), 2011 [23].

- Francisco Bonnin-Pascual and Alberto Ortiz, **An AdaBoost-based Approach for Coating Breakdown Detection in Metallic Surfaces**, *Mediterranean Conference on Control and Automation (MED)*, Corfu (Greece), 2011 [24].

- Francisco Bonnin-Pascual and Alberto Ortiz, **Combination of Weak Classifiers for Metallic Corrosion Detection and Guided Crack Location**, *IEEE International Conference on Emerging Technologies and Factory automation (ETFA)*, Bilbao (Spain), 2010 [25].

- Alberto Ortiz, Francisco Bonnin-Pascual, Andrew Gibbins, Panagiota Apostolopoulou, William Bateman, Marcus Eich, Francesco Spadoni, Massimo Caccia and Leonidas

Drikos, **First Steps towards a Roboticized Visual Inspection System for Vessels**, *IEEE International Conference on Emerging Technologies and Factory automation (ETFA)*, Bilbao (Spain), 2010 [26].

- Francisco Bonnin-Pascual and Alberto Ortiz, **Detection of Cracks and Corrosion for Automated Vessels Visual Inspection**, *International Conference of the Catalan Association for Artificial Intelligence (CCIA)*, Espluga de Francolí, Tarragona (Spain), 2010 [27].

Technical Reports

- Alberto Ortiz, Francisco Bonnin-Pascual and Emilio Garcia-Fidalgo, **Vessel Inspection Assistance by means of a Micro-Aerial Vehicle: Control Architecture and Self-Localization Issues**, Department of Mathematics and Computer Science, University of the Balearic Islands, Tech. Rep. A-02-2013 [28].

Finally, this dissertation does not report on the following publications, which were published during the time as research assistant.

Conference Proceedings and Workshops

- Emilio Garcia-Fidalgo, Alberto Ortiz, Francisco Bonnin-Pascual and Joan P. Company-Corcoles, **Fast Image Mosaicing using Incremental Bags of Binary Words**, *IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm (Sweden), 2016 [29].

- Emilio Garcia-Fidalgo, Alberto Ortiz, Francisco Bonnin-Pascual and Joan P. Company-Corcoles, **A Mosaicing Approach for Vessel Visual Inspection using a Micro-Aerial Vehicle**, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg (Germany), 2015 [30].

- Alberto Ortiz, Francisco Bonnin-Pascual, Emilio Garcia-Fidalgo and Joan Pau Beltran, **A Control Software Architecture for Autonomous Unmanned Vehicles inspired in Generic Components**, *Mediterranean Conference on Control and Automation (MED)*, Corfu (Greece), 2011 [31].

Technical Reports

- Emilio Garcia-Fidalgo, Alberto Ortiz, Francisco Bonnin-Pascual and Joan P. Company-Corcoles, **A Multi-Threaded Architecture for Fast Topology Estimation in Image Mosaicing**, Department of Mathematics and Computer Science, University of the Balearic Islands, Tech. Rep. A-05-2015 [32].

- Emilio Garcia-Fidalgo, Alberto Ortiz, Francisco Bonnin-Pascual and Joan P. Company-Corcoles, **Vessel Visual Inspection: A Mosaicing Approach**, Department of Mathematics and Computer Science, University of the Balearic Islands, Tech. Rep. A-01-2015 [33].

Chapter 2

Related Work

This chapter reviews previous work related with the main issues discussed in this dissertation: robotics applied to inspection tasks and vision-based algorithms for defect detection. These fields, especially aerial robotics, have been growing in the last years, what is reflected in the large amount of papers that have been published. These publications are included in this chapter to provide a nearly full view of what exists at the moment of bringing out this dissertation.

2.1 Robotic Platforms for Inspection

Mobile robotic devices have been widely used for visual inspection. In this regard, the robotics literature contains examples about robots devised for inspecting power transmission lines [34, 35], dams [36, 37], bridges [38, 39], pipes and sewerage [40, 41], aircraft skin [42, 43], etc. In the following sections we will focus on those contributions that deal with key aspects for our research, namely, the use of robots for vessel hull inspection and the use of aerial robots for inspection. It is worth noting that, apart from the publications resulting from our research, there are no contributions about flying robots specifically devised for vessel hull inspection.

2.1.1 Robotic Platforms for Vessel Hull Inspection

The robotics literature contains several contributions about robots for vessel inspection. Most of them consist in underwater vehicles for the inspection of the submerged part of the vessel hull. As a first example, the vehicle presented in [44] is an underwater Remotely Operated Vehicle (ROV) intended for inspection. This is a free-floating vehicle which is able to take paint-thickness measurements of the underwater hull using a specific probe. The *LBC* (Little Benthic Crawler) [45] is another remotely operated robot equipped with a camera suitable for vessel visual inspection. This is a commercial 2-piece robot consisting of a 5-thruster ROV and a removable 4-wheel drive crawler skid assembly. The latter houses a vortex generator which provides attractive force on any relatively flat surface. Another example is [46], where the authors present the design and development of a mechanical contact mechanism that allows an ROV to keep a suitable position and orientation to improve the visual inspection of the hull.

Autonomous Underwater Vehicles (AUV) have the potential for better coverage efficiency, improved survey precision, and overall reduced need for human intervention. Some AUVs devised for vessel inspection are designed to attach and crawl over the hull surface. The *Lamp Ray* [47, 48] is an underwater hull-crawling robot that delivers data on hull plate thickness, form and coating condition. An acoustic beacon positioning system, also known as long-baseline system (LBL), is used for waypoint navigation, providing autonomy. A non-contact underwater ultrasonic (US) thickness gauge and different kinds of probes are used to perform Non-Destructive Testing (NDT), to sense the hull state. The vehicle can operate in free-swimming mode until reaching the hull surface. Then, it holds itself using front-mounted thrusters for suction and moves on wheels over the hull surface, while complex geometry around (e.g. sonar domes, propeller shafts, etc.) is still generally inspected with a free-swimming ROV.

The *AURORA* underwater robot [49] is another hull-crawling robot that can clean a vessel from marine fouling, while simultaneously inspects the state of the hull by means of a US probe and cameras. It can be operated in manual mode and in two different automated modes. In the first one, the robot estimates its movement direction using vision to differentiate the already cleaned areas. In the second autonomous mode, the relative position of the robot is obtained by triangulation using three US sources attached to the vessel hull. Similarly, the *HISMAR* robotic system [50] is conceived to keep the ship hull clean and free of biofouling in order to increase the ship propulsion efficiency. The vehicle is also devised to take plate thickness measurements. Similarly to the *AURORA* robot, the control of the vehicle is provided via an umbilical with power, control lines and hoses used for bringing the cleaning wastes to the surface. Position is estimated by dead-reckoning using optical technology to track the two-dimensional movement over the hull surface. To obtain absolute position estimations, known hull features are used to correct the current tracked position using a magnetic sensing system.

The *HROV* (Hybrid-ROV) [51] is a more recent underwater hull-crawling vehicle devised for the ultrasonic inspection of Floating, Production, Storage and Offloading (FPSO) units. Like the *Lamp Ray*, the *HROV* can also be operated in free-flying mode to reach the hull surface, and then attach itself using the vertical thrusters. Two motorized tracks are then used for the displacement over the hull surface. Its sensor suite includes an altimeter to measure the distance to the hull, a depthmeter, an Inertial Measurement Unit (IMU) providing acceleration and attitude information, and a Doppler Velocity Log (DVL) which provides the hull-relative velocity and position (i.e. dead-reckoning via integration).

Apart from the previous mentioned approaches, most of the contributions regarding underwater robots are based on free-floating platforms which are not attached to the vessel hull. The *CetusII* [52] is a free-floating AUV which uses a specifically designed LBL acoustic beacon system for navigation around ship hulls and similar underwater structures. The vehicle uses altimeters to maintain a constant relative distance from the hull, while the LBL navigation system records its position information along the hull being inspected. This system uses a

transponder net that is deployed over the side of the ship. The inspection of the hull is performed using a forward-looking imaging sonar. This is a high-resolution sonar which is able to create two-dimensional (2D) acoustic intensity images.

Several approaches try to provide solutions for positioning where LBL systems fail (e.g. in environments with extreme multipath effects). The *HAUV* (Hovering AUV) underwater robot [53–55] employs a DVL for hull-relative navigation and control. This sensor is used to lock the AUV onto the ship hull, maintaining distance and orientation, and to compute dead-reckoned coordinates regarding the hull surface. Data provided by an IMU and a depth sensor is also merged for that purpose. A similar configuration of sensors and actuators is used in the *SY-2* [56] and the *REMUS* [57] AUVs. The *REMUS* and the *HAUV* robots are equipped with a dual-frequency imaging sonar which is able to provide images of the vessel hull even in turbid water. The unit installed in the *HAUV* is the Dual-Frequency Identification Sonar (*DIDSON*) [58] which has been also used to create large scale hull mosaics [59].

In [60], the author presents an alternative localization method relying on range measurements taken to surfaces of known curvature, which belong to the vessel hull. This approach, which is intended to be applied to the *HAUV* vehicle, is validated in simulation and using a raft robot.

The *HAUV* is used again in [61]. In this approach, the *DIDSON* sonar is integrated in a Simultaneous Localization and Mapping (SLAM) framework. The latter technique consists in creating an incremental map of an unknown environment while localizing the robot within this map [62]. In [61], the Exactly Sparse Extended Information Filter (ESEIF) algorithm is applied to perform SLAM. This approach needs a manual selection of feature correspondence in the sonar image due to the device's low resolution and low signal-to-noise ratio, in comparison with images taken using optical cameras.

Another SLAM-based approach using the *HAUV* and the *DIDSON* sonar is presented in [63]. This approach consists in aligning point clouds gathered over a short time scale using the Iterative Closest Point (ICP) algorithm. To improve the alignment, the authors present a system for smoothing these "submaps" and removing outliers. Constraints from submap alignment are integrated into a 6-Degrees of Freedom (DOF) pose graph, which is optimized to estimate the full vehicle trajectory over the duration of the inspection task.

Several approaches based on computer vision techniques have been developed in the last decade. In [64] the authors present a system to help ROV operators by minimizing the task of controlling the camera orientation. The system determines the orientation of the hull surface and adjusts the camera position to trace the vessel shape. It consists in three laser pointers and a colour CCD camera mounted in the same pan-tilt unit. The angle between the camera and the surface is calculated by using triangulation of the position of the pixels corresponding to the three laser spots in the camera image.

A stereo-vision system based on mosaic registration methods is presented in [65]. It is integrated in a free-floating commercial ROV to provide the capabilities for positioning,

navigation and mapping during the automated inspection of a ship hull. The authors provide early results for pool and dock trials.

In [66], the authors present an Extended Kalman Filter (EKF) SLAM system using a stereo camera to estimate the position and orientation of an AUV. In their work, they provide laboratory results using a movable measurement apparatus fitted with a stereo camera pointing at the floor, where a printout of a ship hull image is placed.

Hover et al. [67] increase the capabilities of the *HAUV* combining hull-relative DVL odometry, *DIDSON* imaging sonar and monocular camera constraints into a pose-graph SLAM optimization framework to produce an accurate and self-consistent three-dimensional (3D) trajectory estimate of the vehicle. More specifically, they apply sonar and vision-based SLAM processes [68, 69], and combine them via Incremental Smoothing and Mapping (iSAM) [70], to create a single comprehensive map. The resulting vehicle is able to autonomously cover the whole vessel hull, including complex 3D structures as shafts, propellers and rudders.

In [71], the authors improve the vision-based pose-graph SLAM method presented in [69]. They introduce an online Bag-of-Words (BoW) measure for intra and inter-image saliency in order to identify informative key-frames. In a more recent work [72], a similar technique in underwater saliency-informed SLAM is used to relocate the *HAUV* in a multiple session hull inspection. Using this approach, a single-session SLAM result is initially used as a prior map for later sessions, while the robot automatically merges the multiple surveys into a common hull-relative reference frame. To perform the relocalization step, a particle filter is used to leverage the locally planar representation of the ship hull surface. Furthermore, Generic Linear Constraints (GLC) are used to manage the computational complexity of the SLAM system as the robot accumulates information across multiple sessions. The authors provide results for 20 SLAM survey sessions for two large vessels over the course of days, months, and even up to three years.

In [73], a stereo camera and a DVL are combined into a SLAM framework allowing to localize the *HAUV* into a 3D Computer-aided Design (CAD) model of the ship hull. Furthermore, this method labels visually-derived 3D shapes based on their deviation from the nominal CAD mesh. This deviations, which can be caused by biofouling, are added into the prior mesh.

Figure 2.1 shows, by way of example, some of these underwater robots for vessel hull inspection. Other approaches focus on the use of robots magnetically attached to the vessel hull, what makes feasible the inspection above the water line. Despite the fact that some of them are able to estimate their pose (position and orientation), they are basically remotely operated. *SIRUS* [74] and *MARC* [75] make use of magnetic tracks to attach to the dry part of the hull. They both are equipped with US thickness measurement sensors and cameras. *SIRUS* is also able to roughly estimate its position using an EKF to fuse the wheel odometry and the accelerations provided by the IMU. *MIRA* is a fast-deployment lightweight crawler developed within the *MINOAS* project [76–78]. Because self-localization is not feasible in

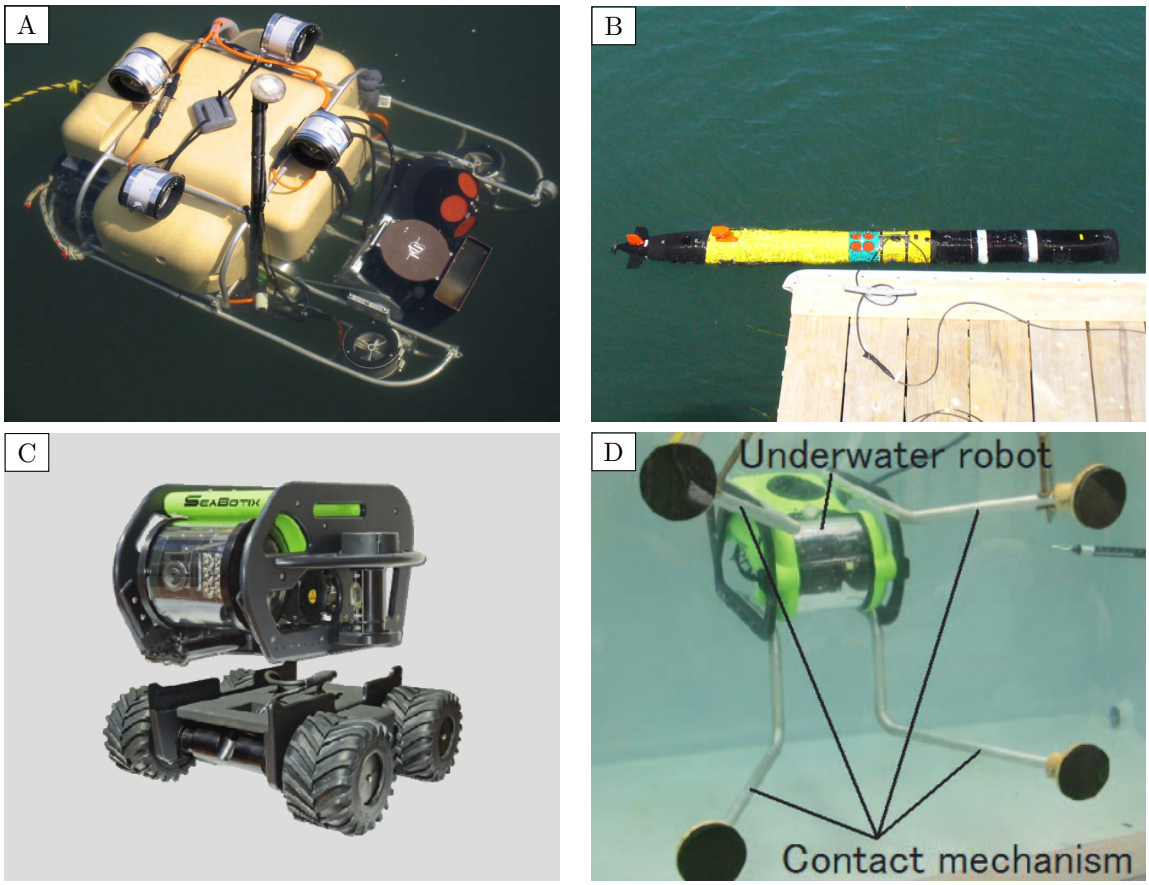


Figure 2.1: Underwater robots for vessel hull inspection: (A) the HAUV autonomous robot equipped with a DVL and a DIDSON sonar [53–55, 61, 63, 67, 71–73], (B) the REMUS 100 AUV [57], (C) the LBC remotely operated robot, comprising a 5-thruster ROV and a 4-wheel crawler skid assembly [45], and (D) the LBV-150 commercial platform equipped with a contact mechanism [46].

such a lightweight vehicle, the position of the robot is estimated using an external 3D tracking system that consists of a camera and a laser range finder mounted on a pan-tilt unit. Regarding submerged hull inspection using magnetic attachment, *SHIV* [79] consists in an underwater crawler provided with 6 magnetic wheels. Similarly, the vehicle presented in [80] is aimed for US-based underwater inspections of FPSO units. Some of these magnetically-attached robots are shown in Fig. 2.2.

Table 2.1 summarizes the main features regarding all the approaches reviewed in this section. They are sorted by year of publication.

2.1.2 Aerial Robotic Platforms for Visual Inspection

The civilian use of Unmanned Aerial Vehicles (UAV) for removing personnel from hazardous situations has grown significantly in recent years. One particular sector of application is

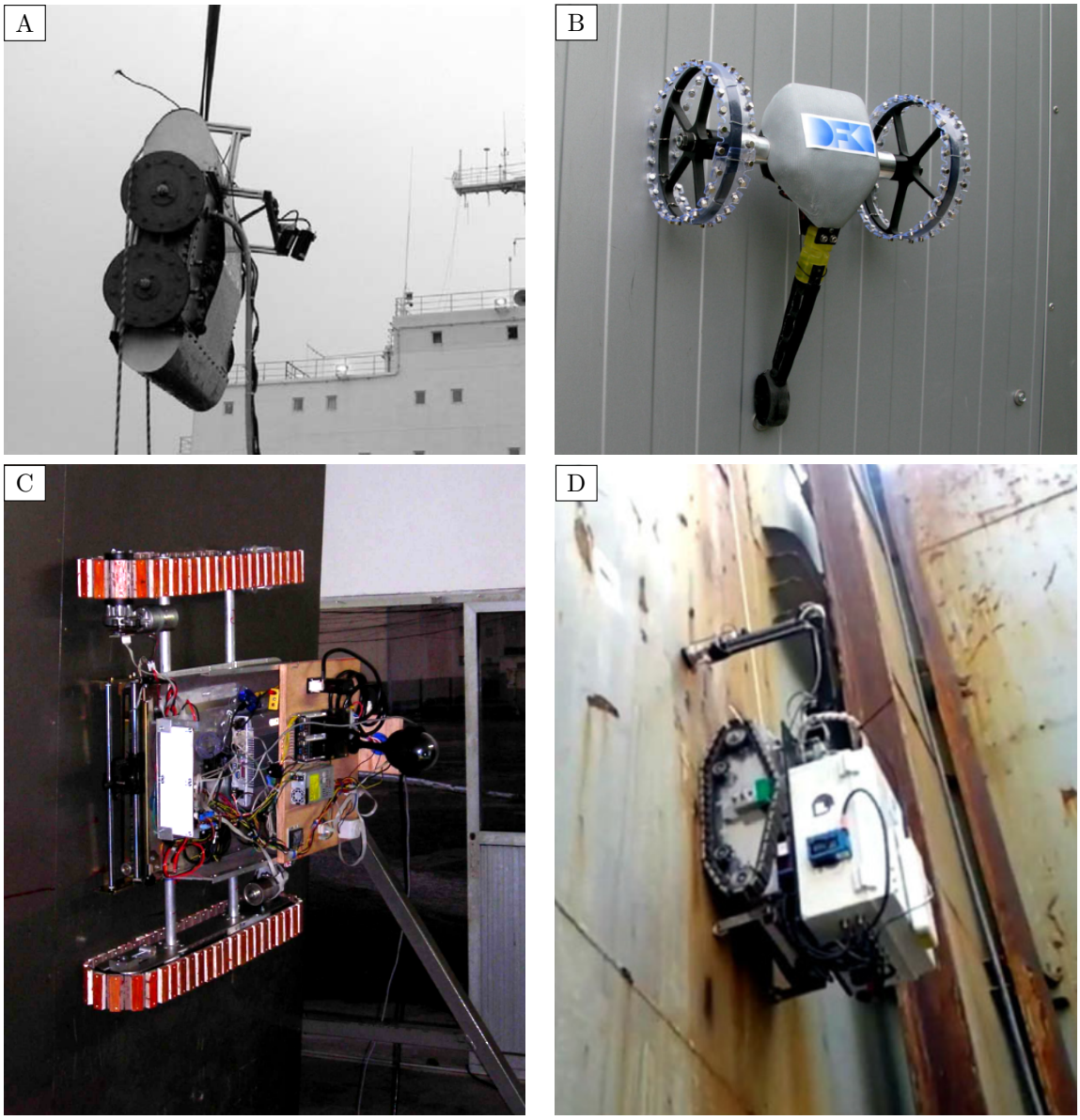


Figure 2.2: Magnetically-attached robots for vessel hull inspection: (A) the remotely operated vehicle presented in [80], (B) the MIRA lightweight crawler [76–78], (C) the SIRUS crawler [74], and (D) the MARC robot equipped with an articulated arm to take thickness measurements of the hull surface [75].

visual inspection. Among the different aerial platform configurations, the Small Unmanned Aerial Systems (SUAS) with capabilities for Vertical Take-Off and Landing (VTOL), such as the multicopters (in the form of quadrotors, hexarotors, octorotors, etc), ducted-fans or coaxial rotor-based helicopters, are the most used platforms. These platforms, sometimes called Micro-Aerial Vehicles (MAV), present high manoeuvrability and are able to operate in confined spaces, including indoor environments. These platforms are typically characterized

Table 2.1: Approaches for vessel hull inspection using robotic platforms.

Ref.	Year	Name	Vehicle	UW	Type	Sensor suite/technique	
						Navigation	Inspection
[79]	1983	SHIV	ROV	Yes	⊗	—	cam./US/mag.
[47, 48]	1999	Lamp Ray	AUV	Yes	⊙/≈	LBL	US/mag.
[44]	1999	—	ROV	Yes	≈	—	mag.
[52]	2002	CetusII	AUV	Yes	≈	LBL+alt.	sonar
[80]	2003	—	ROV	Yes	⊗	—	US
[53–55]	2005	HAUV	AUV	Yes	≈	DVL+IMU+depth.	sonar
[65]	2006	Phantom XTL*	AUV	Yes	≈	stereo odo.	cam.
[60]	2007	—	AUV	Yes	≈	range	—
[49]	2008	AURORA	AUV	Yes	⊙	opt. odo./LBL	US/cam.
[74]	2008	SIRUS	ROV	No	⊗	EKF: IMU+wheel odo.	US/cam.
[61]	2008	HAUV	AUV	Yes	≈	SLAM: sonar	sonar
[56]	2009	SY-2	AUV	Yes	≈	DVL+IMU+depth.	—
[50]	2009	HISMAR	AUV	Yes	⊙	opt. odo.+mag. lmk.	US
[45]	2009	LBC	ROV	Yes	⊙/≈	—	cam.(...)
[57]	2010	REMUS	AUV	Yes	≈	DVL+IMU+depth.	sonar
[76–78]	2011	MIRA	ROV	No	⊗	3D tracker	cam.
[66]	2011	—	—	—	—	EKF SLAM: stereo	stereo
[75]	2012	MARC	ROV	No	⊗	—	US/cam.
[67]	2012	HAUV	AUV	Yes	≈	SLAM: DVL+sonar+cam.	sonar/cam.
[46]	2012	LBV150*	ROV	Yes	△/≈	—	cam./stereo
[51]	2013	HROV	AUV	Yes	△/≈	depth.+alt.+IMU+DVL	US
[71]	2013	HAUV	AUV	Yes	≈	SLAM: cam.	sonar/cam.
[63]	2013	HAUV	AUV	Yes	≈	SLAM: sonar	sonar
[73]	2015	HAUV	AUV	Yes	≈	SLAM: stereo+DVL	sonar/cam.
[72]	2016	HAUV	AUV	Yes	≈	SLAM: cam.	sonar/cam.

Ref: reference number.
Name: * indicates a commercial robot used for testing.
UW: indicates whether the vehicle is underwater or not.
Type: ⊗/magnetic crawler, ⊙/vehicle attached using suction, △/vehicle attached using thrusters and ≈/free-swimming vehicle.
Sensor suite/technique: alt./altimeter, cam./camera, depth./depthmeter, mag./magnetic probe, mag. lmk./magnetic landmark, odo./odometry, opt./optical and US/ultrasound probe.

by a limited payload and autonomy, as well as by small size and reduced cost.

Some scenarios for industrial and generic visual inspection using aerial vehicles are presented in [81], where the platform requirements are discussed as well. Further analysis about UAV properties for visual inspection, focusing on the prevention of image degradation due to the vehicle movement, is presented in [82].

Several approaches provide solutions for visual inspection using teleoperated MAVs fitted with cameras. By way of example, Sampedro et al. [83] present a supervised classification approach for power tower detection and classification in images taken using an aerial vehicle. The same classifier is combined with visual tracking techniques in [84], to track the detected tower across the subsequent images. In [85], a corrosion detector based on colour is used to detect corroded areas in images taken using a MAV. Two different UAVs are used in [86] for

the inspection of photovoltaic plants using colour and thermal cameras. In [87, 88] the visual inspection of bridges using UAVs is addressed. Another example is [89], where an octocopter is used to collect images from building facades. In this approach, the recorded images are stitched together using a mosaicing algorithm, and the final mosaic is analysed to detect the presence of cracks. A more recent approach for building crack detection is presented in [90].

When flying outdoors, MAVs can operate without human intervention thanks to inertial sensors and Global Navigation Satellite Systems (GNSS), such as GPS (Global Positioning System). By way of example, Campo et al. [91] present a system for the autonomous navigation of a low cost quadrotor in open environments, performing a complete coverage of the area. The system is intended for applications such as precision agriculture or environmental monitoring. To perform the navigation, an EKF is used to estimate the vehicle pose, combining GPS and IMU data.

Nevertheless, visual inspections are usually performed in GPS-denied environments where other external positioning systems, such as motion tracking systems, can not be installed. For this reason, aerial platforms for inspection usually must estimate their state (attitude, velocity and/or position) relying on inner sensors and, on many occasions, using on-board computational resources.

The rest of this section tries to provide an horizontal view of the different sensors and techniques applied to the visual inspection using MAVs. We focus on those approaches that go beyond teleoperation and/or pure GNSS-based positioning. The aim is not to be complete, but to show the different trends.

A widely used sensor is the Light Detection and Ranging (LiDAR) device, also known as laser scanner. The use of this sensor, inherited from ground robotics, allows MAVs for positioning (and sometimes mapping), while a camera is typically used for the inspection task. In combination with GPS and IMU data, Serrano [92] proposes using LiDAR data for culvert inspection using a MAV. The idea is to operate the robot outdoors, taking off from a military vehicle, and positioning the MAV in front of the culvert entrance, making an intensive use of GPS data. To perform the inspection inside the culvert, where GPS signal is probably not received, the data provided by the LiDAR sensor is combined with IMU and GPS data within an EKF for the estimation of the MAV state. Then, the operator can use a Pan-Tilt-Zoom (PTZ) camera to perform the inspection.

In [93], a MAV is fitted with a LiDAR and an RGB-D sensor for collaborative mapping of earthquake-damaged buildings. In this approach, the MAV collaborates with two ground vehicles to create a 3D map of the different floors inside the building. In a first stage, a primary ground vehicle is operated to create a 3D map of the environment, using a 3D laser scanner. A secondary ground vehicle is used to carry the MAV to the areas where debris or other obstacles prevent the ground vehicles to keep going. Then, the aerial vehicle is operated through those areas and completes the 3D map. Different laser-based positioning and SLAM algorithms are used to perform the complete mission.

A LiDAR is also used in [94] for positioning and mapping on-board a MAV intended for the visual inspection of equipment and structures in constrained spaces. The data provided by the LiDAR device is merged with IMU data within a particle filter-based implementation of SLAM (FastSLAM 2.0). The vehicle is also equipped with two sonars, one at the top and one at the bottom, to detect upper and lower obstacles as well as to measure distances during the ascending inspection flight and during the take-off and landing procedures. A PTZ camera and several LEDs (Light-Emitting Diode) are used for the visual inspection.

In a more recent work, McAree et al. [95] discuss the development of a semi-autonomous inspection drone capable of maintaining a fixed distance and relative heading to the inspected wall using a LiDAR. The vehicle is operated in semi-autonomous mode so that the pilot can concentrate on the inspection task, while the MAV is in charge of performing the challenging task of distance keeping without pilot input. Within this approach, the authors propose a Model Based Design (MBD) framework to test the distance and yaw controllers in simulation, prior to using them in real world flights.

One of the main drawbacks of using laser scanners in aerial robotics is the relatively heavy weight and elevated power consumption. Recent advances in computational power and CMOS (Complementary Metal-Oxide-Semiconductor) camera technology have made it possible to use computer vision technologies for state estimation on MAVs. Many approaches fuse visual (typically stereo) and inertial data to estimate the vehicle state. An example is [96], which provides a visual-inertial motion estimation system for the visual inspection of industrial environments such as thermal power plant boiler systems. This approach makes use of a sensor comprising an on-board stereo camera augmented with an IMU. On-board this sensor, measurements of linear accelerations and angular velocities are combined with pose measurements in a stochastic coloning EKF. The authors also provide two different strategies for trajectory control which are robust to external disturbances, inaccurate position estimates and delays. While the experiments presented in this paper are performed in a mock environment, [97] shows some results obtained inspecting a boiler system with a more evolved version of the visual-inertial sensor.

Omari et al. [98] propose a navigation system that is built around the commercial version of the previous mentioned visual-inertial system, the *VI-sensor* [99]. This approach estimates both the trajectory of the UAV as well as a 3D map consisting of a sparse set of landmarks. A dense 3D reconstruction can be also generated in post-processing executing the odometry pipeline over all available visual-inertial data.

The *VI-sensor* is combined with two additional CMOS cameras and a LiDAR sensor in [100], for the visual inspection and 3D reconstruction of underground mines. In this approach the MAV is manually operated through the mine to record sensor data. This is post-processed in order to check the feasibility of flying autonomously with the proposed system and sensors. The experiments performed allow concluding that the vehicle has to be protected from dust and water to operate inside mines, what will increase the platform weight and de-

crease its autonomy. Furthermore, due to the lack of a wireless communication system able to operate throughout an entire mine, the vehicle has to be autonomous, detecting problems and deciding by itself which solution it should follow.

Sa et al. [101] present a visual-inertial aided VTOL platform for the visual inspection of pole-like structures, such as light and power distribution poles. The authors present two different approaches for the control system: a Position-Based Visual Servoing (PBVS) using an EKF and an estimator-free Image-Based Visual Servoing (IBVS). An additional contribution is the use of shared autonomy to permit an un-skilled operator to easily and safely perform the inspection using a MAV. The system, which makes use of monocular visual features (lines) and inertial data for the pole-relative navigation, is in charge of maintaining a safe distance and rejecting environmental disturbances, such as wind gusts.

Optical flow techniques have been also applied to visual inspection using MAVs. In [102], an autonomous wall inspection control employing optical flow information provided by a stereo camera system is presented. Using this approach, the inspection velocity along the surface is controlled, as well as the orthogonal distance and the relative yaw angle between the UAV and the observed plane. The authors provide simulation experiments showing the good performance of the control strategy.

In [103], the use of optical flow for inspecting wind turbines and buildings is discussed. The author evaluates two different optical flow methods for local navigation as well as discusses about its application in tracking a moving object and estimating the angular velocity. In this approach, the use of Hough transform is also considered as an alternative to optical flow when there are no features to track. The straight lines detected using the Hough transform are used to find relative angles between blades (in wind turbines) or windows (in buildings), which can be used in the orientation control. While all these techniques are intended to be applied on a hexacopter, they are only evaluated in simulation.

A survey of mobile robots for distribution power line inspection is presented in [34]. It includes different computer vision techniques used on-board UAVs for camera stabilization, pole tracking and automated defect detection. Regarding UAV configuration, two different approaches are reviewed. The first one consists in a ducted-fan rotorcraft [104] which is able to estimate its position and attitude from an image of three conductors of the power transmission line using the Hough transform. The second reviewed approach consists in an autonomous helicopter [105] which is able to fly along power line using a vector-gradient Hough transform for cable detection and stereo vision for determining the position of the cable relative to the helicopter.

Máthé and Buşoniu [106] survey vision and control methods that can be applied to low-cost UAVs intended for visual inspection. Regarding vision-based methods, they overview some techniques for (a) motion tracking and object detection using feature detection/description, (b) motion estimation using optical flow, (c) camera (and vehicle) motion control using visual servoing, and (d) vision-based SLAM. Furthermore, they discuss applications related to

infrastructure inspection and provide some contributions for railway inspection selecting the appropriate vision and control technique to tackle this problem.

Inspection tasks sometimes require physical contact with the inspected surface or structure. In [107], two MAV prototypes for contact-based inspection are presented. The first relies upon a ducted-fan aeromechanical principle, while the second one relies upon a coaxial rotor principle. These vehicles are equipped with a lightweight manipulator, specifically devised to move NDT sensors according to the input provided by the operator, and contact sensors, used to detect physical interactions with the surrounding environment. The human-robot interface make use of a haptic device and augmented reality. Some experiments are reported using a motion tracking system to estimate the MAV state.

Similarly, [108] presents a MAV equipped with a robotic arm attached to the top of the body. The authors discuss the potential of this setup for inspecting structures such as bridges from the underside. This approach presents the dynamic model of the entire system, the non-linear controller implemented, and the first flight experiments performed under a bridge and contacting its surface with a sensor head located at the arm.

A control framework to provide a MAV with physical interaction capabilities is presented in [109]. The authors also provide a contact-based inspection planner which computes the optimal route within waypoints while avoiding any obstacles or other occupied zones on the environmental surface. The resulting MAV is able to perform complex contact-based tasks, e.g. "aerial writing" or interactions with non-planar surfaces. This approach has been validated using pose estimates from a motion capture system, while its performance using on-board sensors (like cameras or LiDARs) has not been evaluated yet.

Also related with contact-based inspection, [110] proposes a high-level control system to allow a UAV to autonomously perform complex tasks in close and physical interaction with the environment. This system combines hierarchical task decomposition, mixed-initiative control and path planning techniques to allow reactivity and sliding autonomy. The approach is evaluated in a physical inspection task and in a visual inspection task, both performed under laboratory conditions.

As happens with the last mentioned approaches, some works focus on issues such as control strategy, task planning or path planning, disregarding the sensor suite and the MAV state estimation. Another example is [111], where the authors propose an MBD framework for planning efficient and robust behaviours for power tower inspection. This approach makes use of reinforcement learning to find an optimal policy to guide a MAV while visiting the target viewing regions along the power tower. The authors provide simulation experiments showing the performance of this framework when the vehicle is flown in the presence of wind gusts and stochastic noise. A last example is [112], which presents a task oriented control strategy for a quadrotor equipped with a robotic arm and a camera attached to its end-effector. This approach describes a hierarchical control law to allow performing visual servoing (primary task) while other tasks (secondary tasks), to minimize gravitational effects or undesired arm

configurations, are also attained. Successful results are presented in simulation.

As far as we know, we presented in 2012 the only approach for vessel visual inspection using a MAV [20]. This consists in a fully-autonomous quadcopter which employs a LiDAR with 30 m range to estimate its position inside the inspected cargo hold. Both odometry and SLAM processes are used for that purpose, while two mirrors are used to deflect part of the laser scans to estimate the distance to the ground and to the ceiling. The vehicle is operated using a "mission description file" which specifies the list of waypoints. This approach assumes that vertical structures that are found in vessel holds are quite similar along their full extent. Navigation and obstacle avoidance is performed in the horizontal plane using the Dynamic Window Approach (DWA) [113]. In [18], we extended this system including a monocular visual odometer using a ground-looking camera. In this approach, the visual odometer is selected instead of the laser-based estimator when the platform performs vertical motion.

Figure 2.3 shows some examples of the reviewed aerial platforms for visual inspection, and Table 2.2 summarizes all the different approaches reviewed in this section. They are sorted by year of publication.

2.2 Vision-based Defect Detection Algorithms

Visual inspection is one of the predominant methods used in quality/integrity assessment procedures. It is a subjective process that relies on an inspector's experience and mental focus, making it highly prone to human error. The development of automated inspection technology can overcome these shortcomings.

Previous approaches on automatic vision-based defect detection can be roughly classified into two big categories. On the one hand, there are lots of contributions on industrial inspection and quality control; that is to say, algorithms that are in charge of checking whether the products that result from an industrial manufacturing process are in good condition. These methods assume a more or less confined environment where the product to be inspected is always situated in a similar position, while lighting conditions are controlled as well. Most of these techniques are collected in [114–117].

On the other hand, several other contributions focus on visual inspection techniques to ensure the integrity of elements or structures that have been subjected to some kind of effort or stress. These methods are typically included in periodical surveys to assess the need of maintenance operations. In this group, which include vessel hull inspection, we can find algorithms for crack detection on concrete surfaces [118], defect detection on bridge structures [119], aircraft surface inspection [42, 120], etc.

The majority of the algorithms from both categories have been devised for the detection of a specific defect on a particular material or surface, while much less methods deal with unspecified defects on general surfaces (some examples are [121–123]). In the following sections, we review existing approaches for crack detection and for corrosion detection. As mentioned

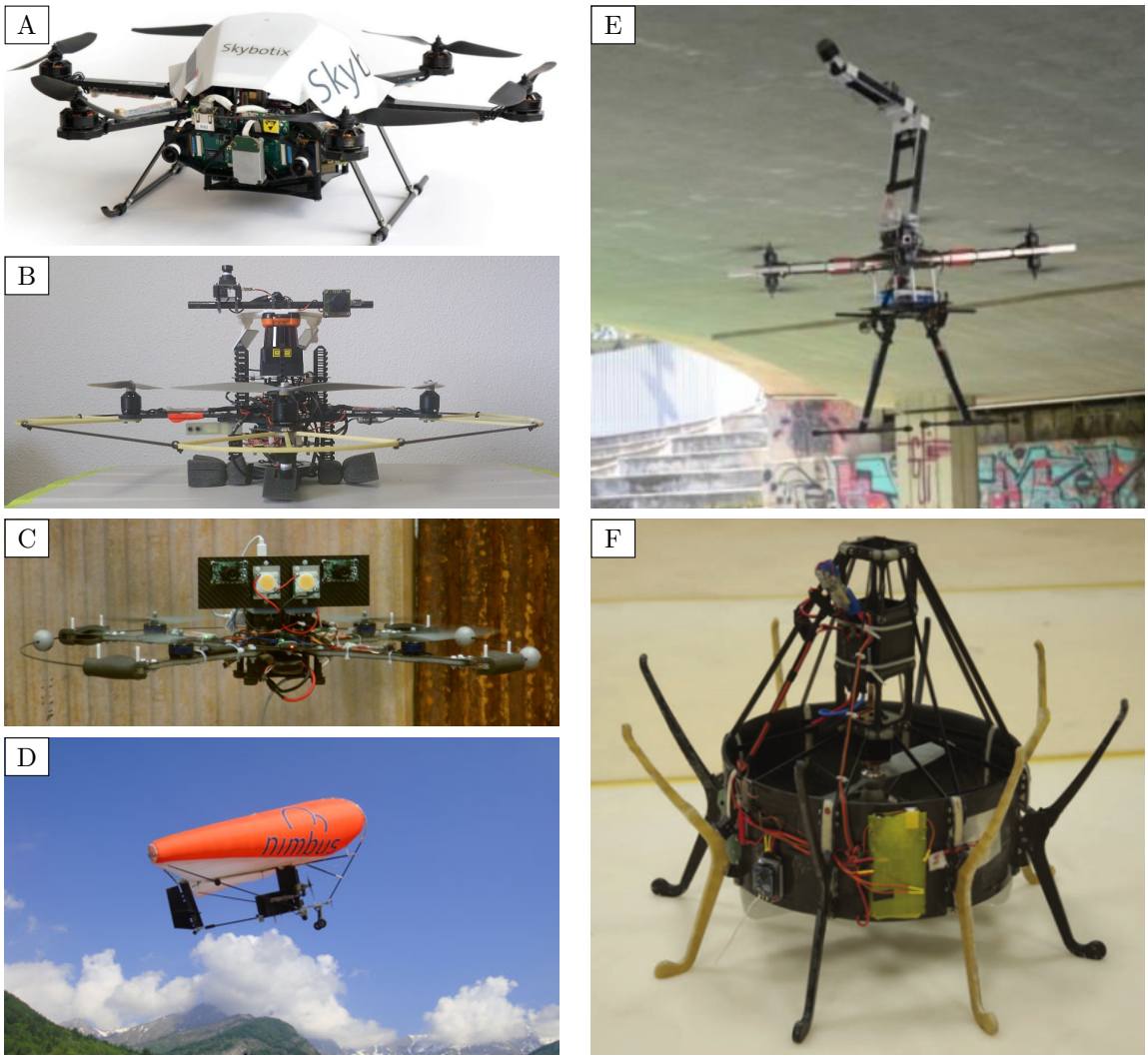


Figure 2.3: Aerial robotic platforms for visual inspection: (A) a hexacopter fitted with the VI-sensor [98], (B) a quadcopter using LiDAR-based SLAM for the visual inspection of vessels [20], (C) a quadcopter fitted with a visual-inertial sensor to inspect boiler systems [97], (D) a hybrid vehicle for the inspection of photovoltaic plants [86], (E) an aerial platform equipped with an arm to inspect structures such as bridges from the underside [108], and (F) a ducted-fan vehicle which allows contact-based inspection [107].

before, we focus on those which solely use digital images as input.

2.2.1 Algorithms for Crack Detection

This section reviews different approaches for vision-based crack detection. An overview of the state of the art is presented in the work by Jahanshahi et al. [119]. This is a survey of image-based techniques for defect detection on bridge structures, including crack detection techniques. In this regard, they consider two different categories: methods based on edge

Table 2.2: Representative approaches for visual inspection using aerial robotic platforms.

Ref.	Year	Inspection	Type	Sensor suite/technique	Output
[105]	2001	Power line	Heli.	st.	img.
[104]	2005	Power line	DF	cam.	img.
[92]	2011	Culvert	4C	EKF: LiDAR+GPS+IMU	img.
[89]	2012	Building facade	8C	—	img.+mosaic+cracks
[93]	2012	Building	4C	SLAM: LiDAR+RGB-D+IMU	3D map
[20]	2012	Vessel str.	4C	SLAM: LiDAR+IMU	img.
[96]	2012	Boiler system	4C	EKF: st.+IMU	img.
[102]	2012	Wall	Sim.	Optical flow: st.+IMU	img.
[107]	2012	Contact	DF/Coax.	IMU, contact sensor	physical inter.
[111]	2012	Power tower	Sim.	—	img.?
[97]	2013	Boiler system	4C	EKF: st.+IMU	img.
[83]	2014	Power tower	—	—	img.+tower
[84]	2014	Power tower	—	—	img.+tower
[86]	2014	PV plant	Hyb./6C	—	img.+thermal
[87]	2014	Bridge	8C	—	img.
[94]	2014	General	4C	SLAM: LiDAR+IMU, 2 US	img.
[98]	2014	General	6C	EKF: st.+IMU	3D recons.
[100]	2014	Mine	6C	EKF: st.+IMU, 2 cam., LiDAR	3D recons.
[103]	2014	WT/Building	6C/Sim.	Optical flow: cam.+IMU+2 US	img.?
[112]	2014	General	4C/Sim.	—	img.
[18]	2014	Vessel str.	4C	SLAM: LiDAR+IMU/vis. odo.	img.
[90]	2015	Building	6C	—	img.+cracks
[101]	2015	Pole-like str.	6C	IBVS/PBVS: cam.+IMU	img.
[106]	2015	Railway	4C	cam.	images+track
[108]	2015	Bridges, etc.	8-4C	—	physical inter.
[110]	2015	Contact	DF/4C	cam./st.+IMU	physical inter./img.
[85]	2016	Metallic str.	4C	—	img.+corrosion
[88]	2016	Bridge	4C	—	img.
[91]	2016	Open env.	4C	EKF: GPS+IMU	img.
[95]	2016	Wall	8C	LiDAR	img.
[109]	2016	Contact	4C	motion tracking	physical inter.

Ref: reference number.
Inspection: PV/photovoltaic, WT/wind turbine, str./structure and env./environment.
Type: Heli./helicopter, DF/ducted-fan, 4C/quadcopter, 6C/hexacopter, 8C/octocopter, 8-4C/ octoquad configuration, Coax./coaxial rotor, Hyb./hybrid and Sim./simulation.
Sensor suite/technique: sensors (and fusion technique) used for pose estimation and/or navigation. st./stereo camera, cam./camera, US/ultrasound range sensor and vis. odo./visual odometry.
Output: img./image, inter./interaction, thermal/thermal image and recons./reconstruction.

detection and methods based on morphological operators. Regarding edge detection, they firstly introduces methods based on the gradient of the image. Among them, the Canny operator [124] is one of the most used, since it provides better results in comparison with other approaches, such as Sobel or Fast-Fourier Transform (FFT) techniques [125]. By way of example, the Canny operator is used in [90] for building inspection using a UAV fitted with a camera. Nevertheless, the authors of this approach indicate that there are parts of the cracks which remain undetected.

In the category of edge detectors, the authors of the survey also include the fast Haar transform, which performs better than the Canny operator in certain scenarios [125], and the method by Siegel and Gunatilake [42]. This is a multi-stage method intended for crack detection on aircraft skin using a robotic crawler. This robot is equipped with a camera and a directional light source to illuminate the inspected area. Crack detection is performed through multi-scale edge detection using a wavelet filter bank. It starts detecting rivet holes, where cracks usually appear, and defining a Region Of Interest (ROI) around them. Edges at different scales are detected using wavelets and then linked through a coarse-to-fine edge linking process. Then, each edge is described using a feature vector containing five different features. Finally, every feature vector is classified using a Neural Network (NN) to determine whether it describes a crack or not. During tests, this detector provides a 72% of accuracy, with a 27% false alarm rate.

The vision literature contains many other crack detectors based on edge searching. In [39], a wheeled robot is used for bridge deck crack inspection and mapping. In this approach, the edge detection is performed convolving the image with a kernel to compute the Laplacian of Gaussian (LoG). This is used to smooth the input image while computing its second derivative. Edges in the resulting image can be found looking for zero-crossings.

Another method based on edge detection is proposed in [89]. It presents a system for building facade inspection using a UAV fitted with a camera. The images collected are stitched together to create a mosaic, which is later analysed searching for cracks. The crack detection method that the authors propose consists in adding Gaussian blur to the original image and then subtracting it form the image again. By doing this step, edges result almost black while the rest of the image results almost white. The authors conclude that the method needs further improvement since small cracks are not very visible after the image processing, whereas man-made edges are misclassified as cracks.

In [126], cracks in concrete surfaces such as bridges, buildings and tunnels are detected using the Histogram of Oriented Gradients (HOG). HOG detects edges by computing the distribution of intensity gradients of the image. Before using HOG, the original gray-scale image is binarized to generate a black and white image. Different results are provided depending on the threshold used during the previous binarization. The authors conclude that further work has to be done to reduce the image noise prior to using HOG.

A more complex method for crack detection on concrete images is presented in [127]. The method includes two preprocessing steps (also used in [128]) and two detection steps. The first preprocessing step is a subtraction process using a median filter to remove slight variations like shadings. In the second preprocessing step, a multi-scale line filter based on the Hessian matrix is used both to emphasize cracks against stains and to adapt the variation of cracks width. In the first detection stage, probabilistic relaxation is used to detect cracks coarsely and to prevent noise. Finally, a locally adaptive thresholding is performed for a finer detection. The complete method attains an Area Under the Curve (AUC) [129] of 0.98, which

is very close to 1, what indicates a very successful detection rate.

Other edge-based crack detection techniques are [130–132]. In general, edge detection techniques provide false positive detections which are produced by the presence of structural member edges or background crack-like objects. In order to minimize them, different filters are used before or after performing the edge detection.

Regarding the use of morphological operators, the survey by Jahanshahi et al. [119] reviews the different basic operators (dilation, erosion, morphological gradient, opening and closing) to end up with two combinations of the opening and closing operators which allow the detection of bright and dark defects, respectively. These operators have been used in [133] to successfully detect cracks in ferrites. Another approach using a similar operator is [134], which presents a method intended for the subway tunnel safety monitoring. This method starts smoothing the gray-scale input image using an average filter. Then all crack-like structures are detected by means of a morphological operator. Image segmentation is then performed employing a thresholding operation and a second morphological operator, which is used to filter out the irrelevant noise. In order to remove the remaining large regional irrelevant objects which are still preserved as cracks, a supervised classification stage is performed. Three different features are used: the standard deviation of shape distance histogram, the number of pixels and the average gray level. The classification is performed using an Extreme Learning Machine (ELM) [135]. This method is selected in this approach because of its universal approximation and classification capabilities. The complete crack detector presents an accuracy above 91%.

Other approaches using morphological operators are [136–138]. In comparison with edge detection techniques, morphological operators do not extract all the edges in the image, which result in less false positive detections. In general, they also generate less noise. Nevertheless, morphological operators require finding the appropriate size and shape of the structuring element to obtain the best detection results.

Apart from the methods based on edge detection/morphological operators, the related literature contains other approaches for detecting cracks in images. Thresholding is a commonly used technique in this field. By way of example, the method presented in [139] starts with two thresholding stages to separate cracks from the background in concrete surfaces. The first thresholding is used to discard clearly non-cracked areas, while the second one tries to find the threshold that maximizes the quotient between the inter-class variance and the inner-class variance, being *crack* and *background* the two classes. This process is followed by a thinning procedure to reduce the crack width to 1 pixel. The remaining pixels are labelled to determine the crack morphology and length. The crack thickness is determined as the number of pixels omitted during the thinning process, while the direction of the crack (regarding the horizontal axis) is computed using a histogram of the directions of the different segments that shapes the crack. A similar procedure can be also performed to compute the thickness of a crack from the thickness of its different segments.

Another methodology for crack detection consists in employing region growing procedures.

In [118, 140], the authors present a crack detection method for concrete surface images using percolation. This is a region-growing procedure based on the natural phenomenon of liquid permeation. The process starts from every pixel (seed pixel) in the image and grows through the darkest neighbouring pixels. The percolation proceeds until reaching a certain initial boundary. Then, the elongation of the percolated area is checked to show whether this is a potential crack (cracks are supposed to be elongated). In that case, the percolation process proceeds iteratively increasing the current boundary and checking the new elongation. Finally, when a previously defined final boundary is reached, the elongation of the percolated area is checked one last time and the seed pixel is accordingly labelled as crack or background. This method improves the classification results achieved by a conventional method that includes wavelet transform and shading correction. Nevertheless, notice that this method uses all the image pixels as seed points for percolation, so that most of the computation time is used to perform percolations starting from background pixels, which are far more than the pixels on cracks. Moreover, since just the seed pixel is labelled at the end of each percolation, every pixel is involved in many percolation processes. In [141], the method is improved to deal with these two issues, adding some additional rules and checks which allow, after every percolation, labelling the entire area as crack/background. The improved method also includes a denoising algorithm, based on percolation as well, to remove the false positive detections. Regarding the original method, these improvements reduce considerably the processing time while increases the classification performance.

A different approach is presented in [142]. It consists in a method for crack detection on asphalt images based on grid cell analysis. The method is devised to deal with the problems of shading (or non-uniform illumination) and strong textures in the images. It consists in dividing the image in cells which are classified as crack or not. A cell is considered a crack if there are two (and just two) pixels of its border which are considerably darker than the others. These two pixels might be the entry and exit points of a crack in the cell. After the first classification, the original image is divided again in overlapping areas and a second classification stage is performed, in order to detect those cracks that coincide with a cell border in the first stage. Finally, a cracked cell verification stage is used to remove false positives caused by strong textures. This consists in checking whether there are dark pixels arranged in a line between the two dark border pixels. The authors report a 13% and 21% of false positive and false negative respectively.

A crack detection approach based on the principle of the grid method is presented in [143]. This method consists in fixing a bidirectional periodical pattern onto the inspected surface and analysing the phase modulation induced by the crack. The Windowed Discrete Fourier Transform (WDFT) is used for detecting the phase of the image with the superimposed pattern. After removing high-frequency variations which result from electronic noise, the discontinuous variations indicate the presence of a crack. Small cracks (5 μm wide) are successfully detected on reinforced concrete beams using this approach. Their localization

accuracy is 1.2 mm, while their opening is measured with a precision of 1 μm. Notice that this approach requires very close-up and controlled capture of images.

Convolutional Neural Networks (CNN) have been also applied to the crack detection problem. In [144], a genetic algorithm is employed to train the weights of a CNN in order to pass through local minima, achieving an average success rate above 90%. Another related approach is [145], which compares the classification performances of a CNN, a Support Vector Machine (SVM) and a Boosting method [146]. The best classification ratios are attained using the CNN.

The use of clustering techniques for small crack detection is proposed in [147]. After an initial thresholding, this method assigns the remaining pixels to clusters of cracks or clusters of background. Then, crack clusters are filtered according to their elongated shapes. The proposed clustering method is aware of whether a point lies in the extension line of an elongated cluster or on one side of it, so that the process can cover the points of another crack fragment separated by a gap while keeping noise points outside.

Notice that the appearance of a crack (length, depth, shape, etc.) can be very different from one surface or material to another. For example, a crack that can be found inside a building after suffering an earthquake is very different to the micro-fissures that sometimes arise in an aircraft wing. Furthermore, the control of the camera-surface distance is crucial to know how big the cracks will appear in the images and, therefore, how to configure the algorithm parameters. In this regard, and unlike previous mentioned methods, [148] deals with the unknown-distance problem presenting a crack detection and quantification method based on depth perception. The drawback of this approach is the need of several pictures of the scene captured from different views. These pictures are used to solve a Structure from Motion (SfM) problem [149]. This procedure provides the structure of the scene as well as the camera's position, orientation and internal parameters for each view. Using the depth perception provided by this 3D reconstruction (i.e. the object-camera distance), a morphological operator is then configured for crack segmentation, also considering the desired crack thickness and the camera parameters. Appropriate features are the extracted from each segmented pattern and used to finally classify real cracks. The performance of a NN, a SVM and a nearest-neighbour classifier are discussed by the authors.

To sum up, a wide variety of computer vision techniques for crack detection have been investigated so far. All of them require a suitable image capture procedure in order to provide good results. This includes a specific distance (typically very short) or camera position regarding the inspected surface. In some approaches, lighting must also be controlled. Furthermore, to provide good results, most of them require from learning and/or parameter-tuning stages. The reviewed approaches are summarized in Table 2.3.

Table 2.3: Representative approaches for vision-based crack detection algorithms.

Approach	Particular technique	References
Edge detection	Canny	[90]
	LoG	[39]
	LoG + labelling + Dijkstra	[131]
	Wavelets	[42]*, [130, 132]
	HOG	[126]
	Image differencing	[89]
	Image differencing + Hessian analysis	[128]
	Image differencing + Hessian analysis + probabilistic relaxation	[127]
Morphological operators		[133, 136–138], [134, 148]*
Other	Region growing	[118, 140, 141]
	Thresholding + thinning + labelling	[139]
	Grid cell analysis	[142]
	Grid method (WDFT)	[143]
	CNN	[144, 145]*
	Clustering	[147]

* indicates that some machine learning technique is applied.

2.2.2 Algorithms for Corrosion Detection

Unlike the case of cracks, the computer vision literature contains just a few contributions for corrosion detection algorithms. Two main features are typically employed for corrosion detection. On the one hand, most of the approaches make use of texture descriptors in order to characterize the roughness of corroded surfaces. On the other hand, colour-based descriptors are also very popular since corrosion typically presents colours ranging from yellow to red.

Some approaches make use of wavelet analysis to describe the texture of corroded areas. A first example is [42]. This approach has been already introduced in the previous section since it describes a robotic device used for aircraft skin inspection, including both crack and corrosion detection. Regarding corrosion, this is detected using the Discrete Wavelet Transform (DWT), which provides a characterization of the image texture at multiple resolutions and orientations. Firstly, a three-level wavelet decomposition is performed, resulting in 10 sub-bands. In a second stage, the image is divided into non-overlapping patches and 10-dimensional feature vectors are computed to describe them. The components of the feature vectors are the energy of the corresponding patch in each of the wavelet transform frames. Finally, each patch is classified as *corrosion* or *corrosion-free* by means of a supervised classification module. To perform the learning stage, a clustering algorithm is used to find the prototype vectors for each class. The classification stage is performed using a nearest-neighbour method. The trained algorithm is able to detect 95% of the corrosion vectors of the test set.

In [150], a similar approach is presented. In this case, the Haar wavelet is used to obtain texture information from the three planes of RGB (Red-Green-Blue) images. In more detail,

each image patch is described using a feature vector which contains the energy and entropy values for the different sub-bands and colour channels. The average luminance of the patch is also added to the feature vector, which results with 25 components. Then, Principal Component Analysis (PCA) [151] is used to reduce the dimensionality of the feature vectors to five components. To classify these vectors as *rust/non-rust*, a training stage is performed using the Least Mean Square (LMS) method.

Similarly, [152] evaluates the effect of using different colour spaces, colour channels and image patch sizes in a colour wavelet-based texture analysis algorithm for detecting corrosion. Like [42], this approach makes use of the DWT to obtain the coefficients that are then used to compute the energy of each image patch. Nevertheless, in [152], this process is applied to the colour channels of the image. Six different colour channel combinations are considered: YCbCr, CbCr, YIQ, IQ, HSI and HS. Notice that CbCr, IQ, and HS combinations result from ignoring the brightness/luminance channel of YCbCr, YIQ and HSI respectively. An NN is trained for the different combinations and considering 10 different patch sizes. The results show that the performance of the detection system improves when the features obtained from the brightness channel are excluded. The colour channel combination which provides the best performance is CbCr, with an AUC of 0.94. The HSI colour space is found the less appropriate for using with the proposed wavelet analysis.

Despite the results presented in [152], HSI (Hue-Saturation-Intensity) and HSV (Hue-Saturation-Value) colour spaces are widely used in colour-based corrosion detectors. These are intuitive models which isolate the brightness information into a single channel. As far as we know, the first approach using HSI colour space for describing corrosion is [153]. For this reason, it is included in this state-of-the-art review, despite the method presented is devised to operate with images captured with a microscope. This method takes 10×10 pixel patches of the different classes and then treats the histograms of each colour channel (H, S and I) as distributions of random variables. After applying the PCA and the varimax [154] approaches, the authors conclude that the mean H value, the mean S value, the median S value, the skews of the S distribution and the skews of the I distribution are appropriate features to be assigned to each patch for classification.

In [155], the authors present an approach for corrosion detection using texture information extracted from the Gray Level Co-occurrence Marix (GLCM) [156]. The GLCM is computed for image patches of both classes, i.e *corrosion* and *non-corrosion*, and different texture descriptors are calculated: contrast, correlation, energy and homogeneity. These descriptors are used to train a Self-Organizing Map (SOM) [157] which performs a clustering process over the different samples, creating several prototypes of both classes. During the classification stage, the nearest-neighbour approach is applied. Results show that 93% of test patches are correctly classified using this method.

Medeiros et al. [158] present a corrosion detector which combines the texture descriptors used in [155] with colour information using the HSI colour space. The colour descriptors

consist in the four first statistical moments extracted from each colour channel histogram. The resulting set of descriptors (texture and colour) is optimized using PCA to eliminate redundant attributes. Finally, the classification is performed using Fisher Linear Discriminant Analysis (FLDA) [159] and different subsets of descriptors. The best results are obtained using 13 features combining both texture and colour information, which provide more than 90% of accuracy.

Some approaches make use of a Support Vector Machine to evaluate the corrosion degree of metallic surfaces. In [160], a SVM is used to classify electric pole crossarms into categories *reuse*, *retire* or *reuse after plating*, depending on the colour of the rust expressed in the RGB colour space. Indeed, this approach compares the performances attained by different machine learning techniques, including a SMV, a k-Nearest Neighbour (kNN), a Radial Basis Function (RBF) network, and a Multi-Layer Perceptron (MLP); but the SVM provides the best results. The method starts reducing the image resolution from 640×480 pixels to 20×15, in order to reduce the number of features. Then, the training and classification stages make use of vectors with 20×15×3 (three colour channels) components, where each colour channel is expressed with a value ranging form 0 to 255. The resulting method provides an accuracy above 97%.

Another approach using a SVM is [161]. It presents a system to categorize images from metallic power transmission towers depending on its deterioration degree. Three classes are defined: *early phase*, *adequate phase* and *late phase*. Two different methods are considered in this approach to represent the colour information which is later used to feed the SVM. The first one consists in using RGB scaling, that is, using a scaled version of the image where the colour of each pixel is computed as the average of the corresponding pixels in the original image. The second approach consists in using the concatenation of the histograms of the three channels in the HSV colour space. Both approaches are assessed using different sizes for the RGB structure or the HSV histogram. The best classification performance provided by the SVM is around 85%, and it is obtained when using an HSV histogram with 192 bins.

A completely different solution is described in [162]. It presents an approach for corrosion detection based on watershed segmentation [163]. This method considers a gray-scale image as a 3D surface where the darkest pixels are the local minima. The segmentation process consists in placing a water source in each local minimum to flood the entire image, building barriers where different water sources meet. These barriers constitutes the watershed segmentation. This method sometimes leads to over-segmentation due to the presence of noise or weak edges in the image. The authors of [162] propose a method to prevent this problem. It consists in merging adjacent regions which present a similar average hue. The resulting segmentations look better despite quantitative results are not provided.

In [164], different pre-processing image enhancement filters are evaluated in order to improve the results obtained with a corrosion detector based on the red channel histogram. The set of filters includes mean filtering, median filtering, Gaussian filtering, wavelet de-noising, Weiner filtering, Bayer filtering, and anisotropic diffusion. The authors propose using the

Peak Signal-to-Noise Ratio (PSNR) to select among the different filters, so that the most suitable one is applied depending on the specific lighting conditions. The results show that the Bayer filter provides the highest PSNR value for the majority of the images.

Recently, a CNN has been applied to the corrosion detection problem. Petricca et al. [165] compare a standard computer vision technique with a CNN for classifying images as *rust/non-rust*. In this approach, an image showing corroded elements is considered as *rust*, despite the rest of the image is showing non-corroded elements or surfaces. The standard technique used for the comparison consists in counting the amount of reddish pixels in the image. The image is considered corroded if the counter exceeds 0.3% of the pixels. On the other side, the CNN is implemented using a pre-trained model based on AlexNet [166]. Results show that the CNN performs better in a real case scenario (78% versus 69% of accuracy). However the authors propose including the standard technique for removing false positives before executing the CNN method.

The only approach which discusses about using a UAV for corrosion detection is [85]. The detection algorithm consists in a simple method using a colour threshold in the HSV colour space. The author provides only qualitative results and indicates that the use of some texture measure probably would improve the performance.

To summarize, almost all the existing approaches for corrosion detection rely on some machine learning technique. This implies that a dataset is needed to perform the training stage and to find the appropriate configuration that provides a good detection performance. Table 2.4 details the main properties of the different approaches for corrosion detection reviewed in this section.

Table 2.4: Approaches for vision-based corrosion detection algorithms.

Method		Ref.	Year	Feat.	Attributes	Learn.	Class.
Machine Learning	Using wavelets	[42]	1997	tex.	Energy	Clust.	Nearest-neighbour
		[150]	2011	col.+tex.	Energy and entropy in col. chan. and mean int.*	LMS	
		[152]	2013	col.+tex.	Energy in col. chan.	NN	
	Using GLCM	[153]	2005	col.+tex. +morph.	Metrics from HSI hist., GLCM and morph.*	Clust.	Nearest-neighbour
		[155]	2009	tex.	Contrast, correlation, energy and homogeneity	SOM	Nearest-neighbour
		[158]	2010	col.+tex.	Contrast, correlation, energy and homogeneity + HSI hist. moments*	FLDA	
	Using just colour	[160]	2005	col.	RGB chan.	SVM	
		[161]	2009	col.	RGB chan. / HSV hist. concatenation	SVM	
		[165]	2016	col.+tex.	HSV chan.	CNN	
WS seg. + merging with mean hue		[162]	2012	int.+ col.			
Red chan. hist.		[164]	2015	col.			
Threshold in HSV		[85]	2016	col.			

Ref: reference number.
Method: WS seg./watershed segmentation, chan./channel, hist./histogram.
Feat.: feature used. tex./texture, col./colour, morph./morphology, int./intensity.
Attributes: attributes used in machine learning techniques. * indicates that PCA is applied.
Learn.: learning process in machine learning techniques. clust./clustering.
Class.: classification process in machine learning techniques.

Chapter 3

An Aerial Robotic Device for Vessel Visual Inspection

As indicated in Section 2.1.2, our first attempt for vessel visual inspection using a MAV was focused on providing a fully autonomous platform [20]. This robotic platform provided successful results in field tests performed in different type of vessels [17]. Nevertheless, the usability of this platform is limited due to the way how inspections are performed. To carry out a mission, this has to be previously specified in a "mission description file" which consists in a list of waypoints. Despite this way of operation is suitable to sweep a vessel surface, e.g. a bulkhead, and take a picture, for example, every half a meter, it is not appropriate to make the vehicle attain a specific point in the vessel structure with unknown coordinates. Furthermore, during field trials, some surveyors demanded the capability of manoeuvring the vehicle with some kind of remote control. Besides, since this autonomous system is based on a position control loop, issues in the position estimate (i.e. due to a malfunction of the laser scanner as for the perception of the surrounding structure) may put the platform in trouble or jeopardize the execution of the inspection mission.

This chapter presents a novel aerial robotic platform devised for the visual inspection of vessels which pretends to overcome the shortcomings of our previous design. Due to the nature of this dissertation, we give special importance to the design decisions, which allow a proper fulfilment of the demanded/desired requirements, while the implementation issues are relegated to second place. The chapter is organized as follows: in Section 3.1, the system requirements are presented, including the requirements needed to accomplish the target tasks and also those necessary to improve the usability of the platform; Section 3.2, overviews the platform, introducing the key aspects of the approach and the operating paradigm; Section 3.3 reviews different sensors that can successfully contribute to a suitable state estimation and/or perception of the environment; in Section 3.4, the control architecture design is detailed; Section 3.5 describes the pipeline that estimates the platform state from the sensor data; Section 3.6 provides the details for the implementation of the MAV; and finally, Section 3.7 reports an extensive evaluation of the platform performance.

3.1 System Requirements

The system requirements have been defined taking into account the target task. The idea is to obtain an aerial robotic device to teleport the vessel inspector through the different structures of the vessel, so that he/she can perceive an appropriate view of the hull state. The requirements to fulfil this task are:

1. the vehicle must allow a close-up view of the inspected surface,
2. the vehicle must obey the commands indicated by the user/surveyor,
3. the vehicle must allow reaching the highest structures of the vessel hull (notice that this requirement is not as obvious at it seems since the robotic platform could be e.g. a magnetic crawler),
4. the vehicle must be able to operate inside the vessel hull, including rather narrow spaces, such as ballast tanks, and
5. the vehicle must be able to operate in dark areas, such as a tanker cargo hold, where daylight can not penetrate.

Other requirements are defined to increase the usability of the platform and/or to reduce the mental workload of the user/surveyor who is performing the visual inspection:

6. the vehicle must implement self-preservation functions such as prevent collisions with the surrounding obstacles, and
7. the vehicle must be operable by non-expert users who maybe have never used a robotic device,
8. the vehicle should provide some autonomous behaviours to alleviate the inspection task to the user/surveyor, especially when performing repetitive operations or those prolonged in time.

3.2 System Overview

The aerial robotic platform has been designed to fulfil the system requirements presented in the previous section. Regarding the vehicle architecture, we have chosen to use a multirotor device. This kind of vehicle, in its different configurations (quadcopter, hexacopter, octocopter, etc.), has been widely used in the recent years for visual inspection tasks, as seen in Section 2.1.2 (see Table 2.2 for a summary). Multirotors present the advantage that they require a simple rotor mechanics for flying control. Unlike single and double-rotor helicopters, multirotors use fixed-pitch blades and the vehicle motion is achieved by simply varying the relative speed of each motor to change the thrust and torque that they produce. Among them, we focus

on those which weigh less than 2 Kg. The reduced size of these MAVs, together with their capabilities for hovering and VTOL, make them suitable for operating in confined spaces and close to structures, which is a crucial feature for being able to achieve close-up visual inspection.

With this aim, the vehicle is equipped with cameras to take high resolution pictures and videos from the vessel hull surface. The inspection in dark spaces, such as ballast tanks or closed cargo holds, is possible thanks to the use of high power LEDs that illuminate the inspected surface. All the pictures are tagged with the estimated pose of the vehicle to perform an effective inspection of the vessel and to allow revisiting the area if necessary. Pose estimation issues are explained in the following sections.

To operate a MAV in close proximity to a structure can be a challenging task due to complex environmental conditions and potentially poor situation awareness of the remote pilot. To reduce the mental workload of the pilot in these situations it is beneficial to give the vehicle its own, artificial, situation awareness. Following this idea, the system architecture has been designed around the Supervised Autonomy (SA) paradigm [167]. This defines a framework for human-robot interactive systems which pretends the alleviation of stress on human users while providing appropriate level of instructions and feedback. In other words, the human user is not burdened with the complete control of the system, so that he/she can concentrate on the task at hand. The SA framework comprises five concepts:

- *Self-preservation*, which includes all the safety aspects of the robot, such as collision and obstacle avoidance. The idea is that all the control related issues fall on the robot.

- *Instructive feedback*, to provide the user the same perception medium as the robot. For example, the system can provide the user images of what the robot sees ahead, or the distance to the nearest obstacles at both sides of the robot.

- *Qualitative instructions*, which are used to command the robot in an easily understood manner. For example, instructions such as “go ahead until you find an obstacle”.

- *Qualitative explanations*, to describe to the user what is happening during the course of a mission using the same language employed for the qualitative instructions. For example, the robot can indicate that is “going forward” or inform about “obstacle detected”.

- *User interface*, which is used to display the instructive feedback and allows the user to give qualitative instructions.

To implement the SA framework, our system has been designed comprising two separate agents. On the one hand, the *aerial platform*, which is fitted with several sensors and actuators, is in charge of all the control-related issues to successfully carry out the specified task. The autonomous controller is also in charge of the self-preservation of the platform. On the

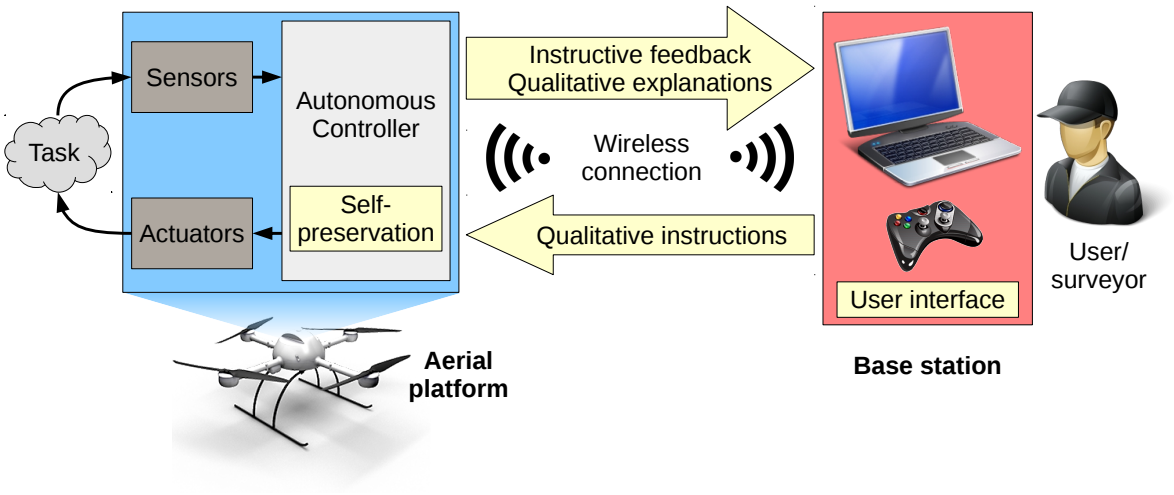


Figure 3.1: Overview of the system based on the Supervised Autonomy paradigm.

other hand, the *base station* is used by the user/surveyor to indicate the qualitative instructions to the aerial platform by means of some input peripheral device. At the same time, the base station is used to provide the user/surveyor with information about the mission's state and the MAV's situation, using instructive feedback and qualitative explanations. The communication between both agents is performed via a wireless connection. An overview of the system is provided in Fig. 3.1.

The vehicle is fitted with a suitable set of sensors to allow the platform to properly estimate its state and perceive its environment under the specific operational conditions that arise inside vessels. In particular, the vehicle can not use GNSS positioning systems due to the lack of line of sight with satellites. Furthermore, the sensor suite must include sensors to allow the vehicle operation in dark spaces, where daylight can not penetrate. Section 3.3 discusses about the different sensors that have been considered to be installed on-board the MAV.

The autonomous controller comprises a set of behaviours which are in charge of accomplishing the specified task while ensuring the platform self-preservation. For example, a behaviour is in charge of moving the platform as indicated by the user, another prevents collisions with the surrounding obstacles, another keeps a constant distance with the inspected surface, etc. The different behaviours developed are detailed in Section 3.4.4.

This design introduces the user/surveyor in the position control loop, allowing him/her to take the platform to the desired point while being assisted at all times by the control software. Furthermore, waypoint navigation is not used in this design and, hence, position estimation is not required for the control system. The approach adapted is based on a velocity controller, what requires proper speed estimations.

3.3 Sensor Suite

The design of our robot requires ensuring accurate estimation of speed (for the control software), as well as a position estimate (not so critical) to tag the pictures taken during a flight. A detailed description of the platform state, including the estimated velocity and position, is provided in Section 3.4.2.

As mentioned before, the need of flying inside closed spaces make impossible the use of GNSS systems such as GPS. Furthermore, the large dimensions of the holds and tanks inside vessels, together with the presence of traces of goods or rust particles, make unfeasible the use of motion tracking systems. Because of that, the vehicle state estimation must rely on on-board sensors. Among them, we focus on lightweight devices that can be carried as payload by small UAVs.

All MAVs are normally equipped with an IMU. This device usually comprises three accelerometers, three gyroscopes and a magnetometer. Using these sensors, the IMU can estimate the accelerations of the vehicle in the three axes (longitudinal, lateral and vertical), the three angular velocities around these axes, and the attitude of the platform. Despite they are widely used, IMUs can not be employed alone to estimate the platform velocity or position. Linear velocities are sometimes computed by integrating the accelerations measured with the IMU, but this just works for a short period of time (maybe a few seconds). Then, the inexactitudes in the acceleration measure, together with the effect introduced by the finite sampling frequency of the sensor, make the velocity estimation degenerate. For this reason, to obtain a proper velocity or position estimation, IMU data have to be combined with information provided by other sensors.

In the following, we consider and perform a qualitative analysis of different sensing options:

- Range sensors provide a measure of distance to some surface or obstacle. The maximum and minimum detection range depends on the size of the obstacle and the kind of sensor. On the one hand, US range sensors usually operate up to 5-6 m, despite some devices can detect obstacles a bit farther. The aperture of the US beam also depends on the specific device, but typically varies between 0.5 and 2 m at maximum range, while its resolution can vary from 1 mm to 1 inch. These sensors are widely used in MAVs for obstacle detection due to its low price (between 30 and 50 €) and weight (around 5-6 g). Figure 3.2 shows two examples of US range sensors used in MAV applications.

- On the other hand, optical range sensors comprise several kinds of devices depending on whether they are based on infrared light (IR) or laser. The maximum detection distance is typically farther than the US maximum range. In the small devices suitable for MAVs, this can be above 40 m (only the laser-based sensors). Since they provide the distance to a single point, they are not usually used for obstacle detection but to measure distances to large surfaces such as walls or to the floor. In comparison with US-based sensors, they present a higher update rate, they are a bit heavier (starting

Figure 3.2: US range sensors: (left) Parallax PING))) Ultrasonic Sensor and (right) Maxbotix XL-Maxsonar-EZ4.

Figure 3.3: Optical range sensors: (left) Teraranger One and (right) Lidar-Lite v2.

from 8 g) and more expensive (starting from 150 €). Figure 3.3 shows two examples of optical range sensors widely used in MAV applications. The sensor on the left is an IR time-of-flight sensor, while the device on the right is a laser-based sensor. Finally, notice that the range provided by any distance sensor can be also used, by means of differentiation, to compute a velocity for a certain direction when traveling along that direction and a static surface is on the platforms way.

- Laser scanners provide the distance to the surrounding objects in a wide angular range (from 180° to 270°, depending on the device), with a maximum detection distance that varies between 1 and 40 m. These are widely used in MAV applications for motion estimation [20, 92, 95] and SLAM [20, 93, 94, 100]. Since laser scanner provides 2D information, the MAV must be fitted with some additional sensors to obtain a 3D perception

Figure 3.4: Laser scanners: (left) Hokuyo UTM 30LX and (right) Hokuyo UST 20LX.

of the environment (see for example [93]) or, when flying in indoor environments, to assume that the obstacles (walls) are completely vertical (as assumed in [20]). Despite having been mitigated along the last years, the main drawback of laser scanners is still its weight. The devices typically used in MAV applications weigh between 130 and 370 g. Its price is still high, ranging between 1200 € for a sensor with a range limited to 4 m, and 5000 € for a 30 m range device. Figure 3.4 shows two different laser scanners from Hokuyo, which is one of the main manufacturers of LiDARs used in MAV applications.

- Unlike laser scanners, RGB-D sensors provide a 3D perception of the environment. An RGB-D sensor typically comprises two cameras and an IR emitter. The first camera is an IR camera used to capture the predefined dotted pattern projected by the IR emitter. Shifts in this dotted pattern determine the depth of the region. Therefore, the sensor provides a 3D point cloud. The second camera is an RGB camera used to provide colour information. RGB-D sensors are cheaper than laser scanners, with a price starting from 90 €. Nevertheless, the maximum perception range of RGB-D sensors is much more limited. It varies from 1.2 to 5 m, depending on the device. Thus, RGB-D sensors are suitable for indoor environments (see for example [93, 168]), but useless in wide environments if we do not operate close to walls/structures. The weight of RGB-D sensors fitted on-board MAVs ranges from less than 100 g to more than 1 kg (these must be lightened before installing). Figure 3.5 shows two examples of RGB-D sensors. The sensor on the left is the Microsoft Kinect sensor, which is one of the most used RGB-D devices. The sensor on the right is a more recent lightweight device from Intel corporation.

- The use of vision-based methods for state estimation and/or mapping using MAVs is nowadays pushed by the new powerful processors which are fitted in small and lightweight computers. Some approaches make use of vision systems based on monoc-

Figure 3.5: RGB-D sensors: (left) Microsoft Kinect and (right) Intel RealSense R200.

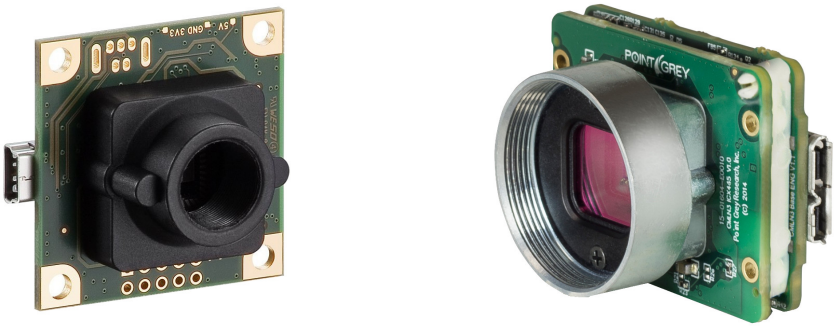


Figure 3.6: Cameras used in MAV applications: (left) IDS uEye UI-1221LE and (right) PointGrey Chameleon3.

ular cameras for position estimation [18], velocity estimation by means of optical flow computation [103, 169, 170], visual servoying [101] or SLAM [171–174]. Since depth perception can not be obtained when using a single camera, all this approaches make use of some additional sensor/technique for that purpose. Other approaches make use of a stereo rig with two cameras so that the depth information can be obtained by triangulation. In this regard, the related literature contains examples for stereo visual odometry [97,98,100,175], stereo optical flow [102] and stereo SLAM [176]. The accuracy and resolution of a stereo-based system depends, among others, on the distance between the two cameras, which is called the baseline. All camera-based systems require light and textured environments to work properly. Therefore, the main drawback of these systems is that they can not be used in dark or poorly-textured environments. Regarding the hardware, several manufacturers produce high-resolution, small and lightweight cameras suitable to be installed on-board MAVs, with prices usually ranging between 300 and 700 €. Some examples can be found in Fig. 3.6.

Table 3.1 summarizes the qualitative analysis of the reviewed sensors. Notice that different sensor configurations are possible to estimate the same platform state. Based on that, we propose three different sensor suites to be installed on our platform, which are enumerated next.

Table 3.1: Qualitative analysis of sensors susceptible to be used in MAV applications.

Sensor	Measure	Lightweight	Long range	Low price
US range sensor	Distance to obstacle	* * * *	*	* * * *
Optical range sensor	Distance to surface (wall, floor, etc)	* * *	* * * *	* * *
Laser scanner	Distance to surrounding obstacles in 2D	*	* * * *	*
RGB-D sensor	Perception of the environment in 3D	* *	*	* * *
Monocular camera	Odometry, optical flow, SLAM, detection, etc.	* * * *	—	* * *
Stereo camera	Odometry, optical flow, SLAM, detection, etc.	* * *	* * *	* *

Sensor suite 1

It is devised for small UAVs with a very limited payload. It is based on the use of velocity estimates regarding the floor and/or the front wall (the wall under inspection). These estimates are obtained using optical flow sensors which provide the velocity combining a camera and a US range sensor. The camera is used to compute the optical flow while the range sensor is used to introduce the scale to the flow measures, to obtain speed values, as well as the distance to the wall. Two additional US range sensors are used to detect the obstacles at both sides of the platform. The height estimation is performed using an optical range sensor with a larger maximum range than the US sensor fitted in the down-looking optical flow sensor. Finally, the pose of the platform is estimated using a forward-looking camera which provides images to feed a monocular SLAM algorithm. To summarize, the first sensor suite comprises:

- an IMU for attitude estimation,
- an optical flow sensor, comprising a camera and a US range sensor, looking forward,
- an optical flow sensor, comprising a camera and a US range sensor, looking downward,
- an optical range sensor looking downward,
- a US range sensor looking to the left,
- a US range sensor looking to the right and
- a colour camera looking forward.

Since this sensor suite is based on the use of several cameras, it requires a sufficiently illuminated scene together with the presence of distinguishable points (known as features),

to allow a proper estimation of the vehicle displacement and position. In other words, this lightweight sensor suite is not suitable for dark environments.

Sensor suite 2

It is suitable for platforms with a larger payload. It is based on the use of a laser scanner for velocity estimation, obstacle detection and position estimation via SLAM. Regarding the first sensor suite, the optical flow and range sensors are removed, as well as the forward-looking camera for displacement estimation. The second sensor suite comprises:

- an IMU for attitude estimation,
- a laser scanner and
- an optical range sensor looking downward.

The use of a laser scanner makes feasible the operation in dark or poorly textured environments, but requires the presence of, from time to time, changes in the structure, for a proper estimation of the MAV displacement. For example, this sensor is affected by the so called "canyoning" effect, i.e. the miss-estimation of the displacement along a corridor or canyon because of lack of structure in the walls.

Sensor suite 3

It is intended to provide a more robust system suitable for flying in a larger variety of environments. It results from combining the first and second sensor suites so that both the optical flow sensors and the laser scanner are used to estimate the velocity and position. The laser scanner is used as the main sensor, while the information provided by the optical flow sensors allows a suitable estimation in non-structured environments or corridors, preventing miss-estimations such as the ones produced by the "canyoning" effect. To summarize, this last sensor suite comprises:

- an IMU for attitude estimation,
- an optical flow sensor, comprising a camera and a US range sensor, looking forward,
- an optical flow sensor, comprising a camera and a US range sensor, looking downward,
- a laser scanner and
- an optical range sensor looking downward.

The sensor suites will be referred to as SS1, SS2 and SS3 from now on. The way how the data provided by all the sensors is processed and combined to estimate the platform state is detailed in Section 3.5. Notice that SS2 and SS3 require an additional camera to perform the visual inspection of the vessel. This has not been included in the previous lists since it is not necessary for the estimation the platform state.

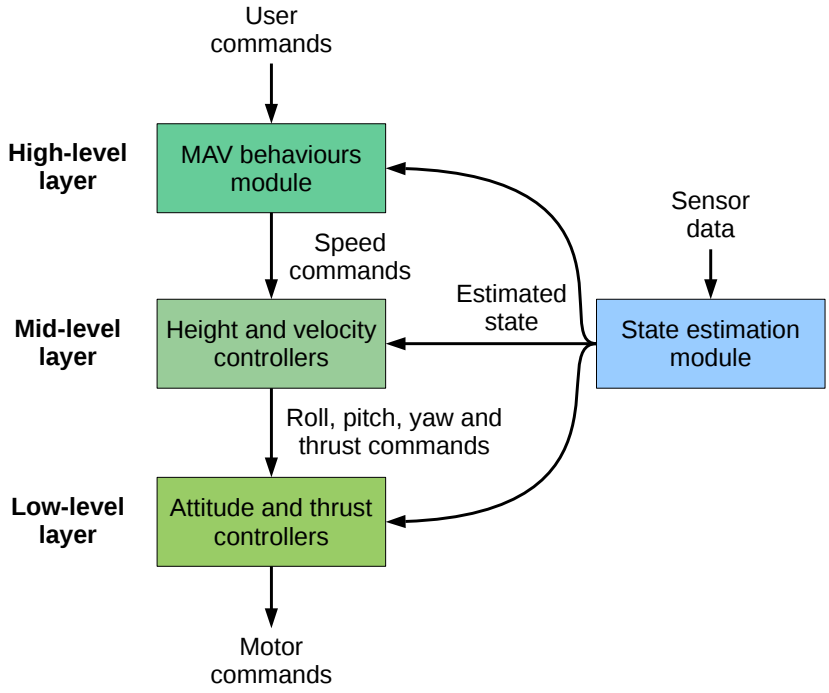


Figure 3.7: Control architecture.

3.4 Control Architecture

The control architecture has been designed as a layered structure, so that each layer corresponds to a different control level. This architecture is shown in Fig. 3.7. The lowest layer of the architecture comprises the attitude and thrust controllers which provide the motors commands, while the mid-level layer consist of the height and velocity controllers. These two layers are detailed in Section 3.4.3. The high-level layer is in charge of executing the *MAV behaviours* module comprising several robot behaviours. These behaviours collaborate to provide the velocity commands to the middle layer. Notice that the control software has been designed following the SA paradigm, so that this last layer is in charge of the platform *self-preservation* and the fulfilment of the *qualitative instructions* given by the user/surveyor, as explained in Section 3.2. A description of all the behaviours and the way how they interact with each other can be found in Section 3.4.4.

Apart from the control layers, the *State estimation* module is in charge of processing and combining all the sensor data to estimate the platform state. The state estimate is used by the different control layers as seen in Fig. 3.7. The complete pipeline executed in this module is detailed in Section 3.5.

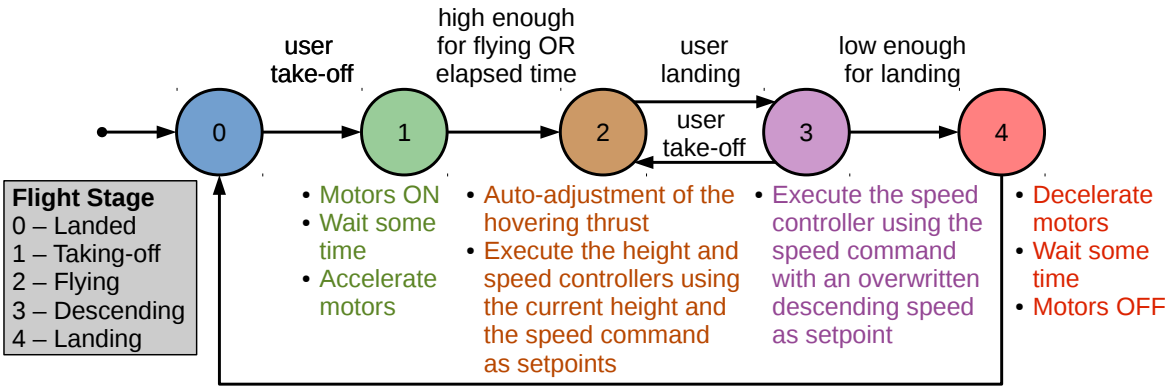


Figure 3.8: Flight control state machine.

3.4.1 Flight Stages

The MAV flight control is implemented as a finite state machine that comprises five states: *landed*, *taking-off*, *flying*, *descending* and *landing*. The transitions between the states take place when the particular conditions are met. For example, the vehicle changes from *landed* to *taking-off* when the user starts the take-off manoeuvre from the user interface. Then, the motors start running and perform an acceleration ramp to elevate the vehicle from the floor. Some other transitions do not depend on the user commands but on sensor data and on the vehicle state. For example, the vehicle changes from *taking-off* to *flying* when the estimated height is above a certain value or after some time at a high level of motor thrust. The complete state machine is shown in Fig. 3.8.

When the vehicle is in the *flying* stage, three controllers are in charge of tracking the speed command in the longitudinal, lateral and vertical axes. When the speed command in the vertical axis is zero, the height controller is enabled to provide the suitable command to the vertical speed controller in order to keep the current height. Furthermore, when the vehicle enters the *flying* stage for the first time, an auto-adjustment of the hovering thrust is performed to stablish the suitable value for the specific air conditions. Details about the hovering thrust and the speed/height controllers are provided in Section 3.4.3.

The landing procedure is split into two stages. When the user starts the landing manoeuvre, the vehicle changes to the *descending* stage. Within this stage, the three speed controllers are still active and the vertical speed command is overwritten by a descending speed in order to reduce the flight height. The longitudinal and lateral commands are fed with the setpoint indicated by the MAV-behaviours module, as in the *flying* stage, so that the platform still obeys the user commands, prevents collisions, etc. When the platform is close enough to the floor, it changes to the *landing* stage, which performs a deceleration ramp of the thrust and, finally, switches the motors off. The user can cancel a landing manoeuvre during the

descending stage by starting a take-off manoeuvre. In that case, the vehicle changes back to the *flying* stage.

3.4.2 Platform State

The MAV state components are defined following the ROS coordinate frame conventions [1] (see Section 3.6.2 for a brief description of ROS). These define the axes orientation in relation to a body as:

- X - forward, in the longitudinal axis
- Y - left, in the lateral axis
- Z - up, in the vertical axis

Regarding rotations, the right hand rule is followed to define the positive direction with regard to the previous axes. Thus, roll (φ), pitch (θ) and yaw (ψ) angles are defined with regard to X, Y, and Z axes respectively. Figure 3.9 shows these coordinate frame conventions.

The pose of an object is determined by its position (x, y, z) and orientation (φ, θ, ψ). Regarding the object's position, it is determined as the translation from the world origin to the final location of the object center, with regard to the world coordinate frame. The orientation is given using the Euler angles, which are applied in the order yaw-pitch-roll (Z-Y-X).

Linear velocities ($\dot{x}$, $\dot{y}$, $\dot{z}$) and accelerations ($\ddot{x}$, $\ddot{y}$, $\ddot{z}$) are defined with regard to the body fixed coordinate frame of the moving object. The same coordinate frame is used to provide the angular velocities ($\dot{\varphi}$, $\dot{\theta}$, $\dot{\psi}$).

After setting all the coordinate frame conventions, the state of the MAV can be defined. The height and velocity controllers, in the mid-level control layer, require the corresponding estimates of height (z) and linear velocities ($\dot{x}$, $\dot{y}$ and $\dot{z}$). Furthermore, the velocity controllers also require the linear accelerations ($\ddot{x}$, $\ddot{y}$ and $\ddot{z}$) to compute the roll, pitch and thrust commands, as explained in Section 3.4.3. Similarly, the orientation of the platform (φ, θ and ψ) and the angular velocities ($\dot{\varphi}$, $\dot{\theta}$ and $\dot{\psi}$) are required by the attitude controllers in the low-level control layer to compute the motor commands.

Regarding the high-level control layer, the MAV behaviours module requires the distance to the obstacles situated below (d_b), in front (d_f) and at both sides of the platform (d_l and d_r, for left and right respectively). The height estimation z is performed with regard to the take-off location, and may not coincide with the distance to the nearest obstacle situated below the platform d_b (see 3.5 for details).

Finally, the full position of the platform with regard to some agreed coordinate origin (typically the take-off location) is required to tag the pictures taken during a inspection mission. Thus, x and y estimates are also required. The full MAV state is defined in Table 3.2, indicating which module or control system requires each state variable.

[1] http://www.ros.org/reps/rep-0103.html

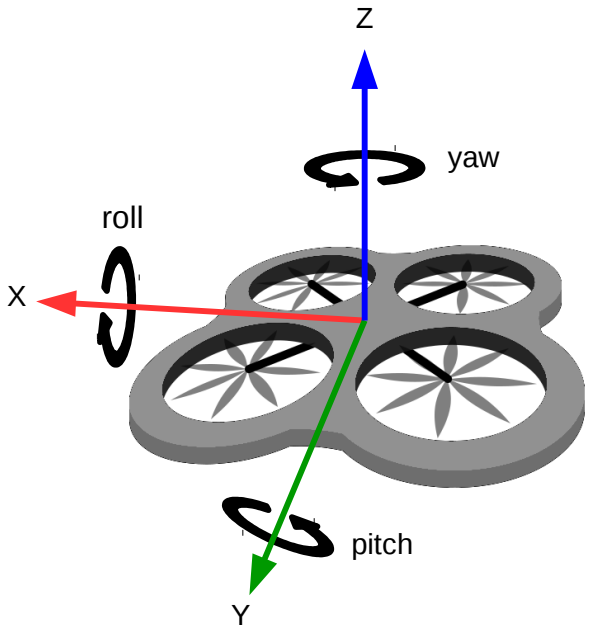


Figure 3.9: Coordinate frame conventions.

3.4.3 Flight Control

This section focuses on the low- and mid-level control layers. The low-level layer is in charge of executing the attitude and thrust controllers, i.e. this layer comprises the controllers in charge of keeping the desired roll (φ_d), pitch (θ_d), yaw velocity ($\dot{\psi}_d$) and thrust ($\mathcal{T}_d$). Since these controllers are typically provided by the manufacturer as part of the platform firmware, they are not further discussed in this dissertation.

The mid-level control layer is in charge of tracking the desired linear velocity commands $\dot{x}_d$, $\dot{y}_d$ and $\dot{z}_d$ by providing the suitable attitude and thrust commands to the low-level controllers. Notice that, when a multirotor is tilted (rotated around the X and/or the Y axis), it suffers an acceleration towards the specific direction. Therefore, the movement along the X axis can be controlled providing the suitable pitch commands, while roll commands are used to control the movement along the Y axis. Regarding movements along the Z axis (changes in height), they can be controlled by by means of a suitable thrust.

The three linear velocity controllers have been implemented as Proportional-Integral-Derivative (PID) controllers [177]. Given an error value $\mathcal{E}(t)$ as the difference between a desired setpoint and a measured process variable, the output signal $u(t)$ of a PID controller is the weighted sum of three different terms:

- a proportional term which depends on the current error $\mathcal{E}(t)$,
- a derivative term which depends on the derivative of the error with respect to time

Table 3.2: MAV state. The first column indicates the modules or control systems which require each state variable.

	x	y	z	φ	θ	ψ	$\dot{x}$	$\dot{y}$	$\dot{z}$	$\dot{\varphi}$	$\dot{\theta}$	$\dot{\psi}$	$\ddot{x}$	$\ddot{y}$	$\ddot{z}$	d_b	d_f	d_l	d_r
Attitude contr.				×	×	×				×	×	×							
Velocity contr.							×	×	×				×	×	×				
Height contr.			×						×										
MAV behaviours																×	×	×	×
Image tagging	×	×	×			×													

Position: x, y and z.
Orientation: φ, θ and ψ.
Linear velocities: $\dot{x}$, $\dot{y}$ and $\dot{z}$.
Angular velocities: $\dot{\varphi}$, $\dot{\theta}$ and $\dot{\psi}$.
Linear accelerations: $\ddot{x}$, $\ddot{y}$ and $\ddot{z}$.
Distances to obstacles below, in front and at both sides: d_b, d_f, d_l and d_r

$\partial\mathcal{E}(t)/\partial t$, and

- an integral term which depends on the accumulated error $\int_0^t \mathcal{E}(\tau)\, d\tau$,

so that the PID output can be written as

$$u(t) = K_p\, \mathcal{E}(t) + K_d\, \frac{\partial \mathcal{E}(t)}{\partial t} + K_i \int_0^t \mathcal{E}(\tau)\, d\tau, \tag{3.1}$$

where K_p, K_d and K_i, all non-negative, denote the tuning coefficients for the proportional, derivative, and integral terms. If K_d and/or K_i are set to zero, different controller configurations can be set: P, PI, PD and PID. The proportional term is always required. It is the part responsible of reducing the error: the bigger the error, the stronger the control signal. However, a purely proportional controller usually causes severe overshoot, leading to strong oscillations, and never counteracts completely the static error. The derivative term has the effect of dampening the oscillations: the higher the rate of change (the derivative), the more this term contributes to slow down this rate of change, reducing the overshoot and the oscillations. Finally, the integral term is responsible of eliminating the steady-state error: i.e. in a biased system requiring a constant control input to hold a state, a pure PD controller will settle above or below the setpoint. Using the accumulated error, the integral term compensates this bias. Nevertheless, the integral of the error needs to be saturated to prevent an excessive effect (which produces strong oscillations) when the error is still large.

In the case of our MAV, the PIDs for controlling the three linear velocities can be written as

$$\theta_d(t) = K_p^{\dot{x}}\, \mathcal{E}_{\dot{x}}(t) + K_d^{\dot{x}}\, \frac{\partial \mathcal{E}_{\dot{x}}(t)}{\partial t} + K_i^{\dot{x}} \int_0^t \mathcal{E}_{\dot{x}}(\tau)\, d\tau, \tag{3.2}$$

$$\varphi_d(t) = K_p^{\dot{y}}\,\mathcal{E}_{\dot{y}}(t) + K_d^{\dot{y}}\,\frac{\partial \mathcal{E}_{\dot{y}}(t)}{\partial t} + K_i^{\dot{y}}\int_0^t \mathcal{E}_{\dot{y}}(\tau)\,d\tau, \tag{3.3}$$

$$\mathcal{T}_d^*(t) = K_p^{\dot{z}}\,\mathcal{E}_{\dot{z}}(t) + K_d^{\dot{z}}\,\frac{\partial \mathcal{E}_{\dot{z}}(t)}{\partial t} + K_i^{\dot{z}}\int_0^t \mathcal{E}_{\dot{z}}(\tau)\,d\tau, \tag{3.4}$$

where the outputs θ_d, φ_d and $\mathcal{T}_d^*$ are the desired pitch, roll and thrust, $\mathcal{E}_{\dot{x}}$, $\mathcal{E}_{\dot{y}}$ and $\mathcal{E}_{\dot{z}}$ are the errors in the three linear velocities, and K_p^{DOF}, K_d^{DOF} and K_i^{DOF} are the constants for the proportional, derivative and integral terms for the corresponding degree of freedom, namely $\dot{x}$, $\dot{y}$ and $\dot{z}$.

Notice that the desired thrust resulting from the PID ($\mathcal{T}_d^*$) must be added to the thrust value necessary to compensate the weight of the platform, i.e. the thrust for hovering, $\mathcal{T}_h$. Thus, the final desired thrust value is obtained as

$$\mathcal{T}_d(t) = \mathcal{T}_h + \mathcal{T}_d^*(t). \tag{3.5}$$

The suitable value for the hovering thrust depends, obviously, on the vehicle weigh, but also on the air density, which varies with the air temperature. For a proper configuration of this value, an auto-adjustment procedure is performed the first time that the MAV takes off. After changing to the *flying* stage, the MAV height is checked to be high enough to prevent perturbations due to the proximity to the ground. Then, the vehicle is left to free hover and the mean of the desired thrust is computed. During this process, the height controller will try to contribute to the initial $\mathcal{T}_h$ the suitable value to hover. After some seconds, $\mathcal{T}_h$ is overwritten with the computed mean, but limiting the update $\Delta\mathcal{T}_h$ to $\mathcal{T}_{h\_incr}$, to prevent oscillations. The process is repeated until $\mathcal{T}_h$ converges. This auto-adjustment procedure is also detailed in Alg. 3.1.

As mentioned before, the error in the linear velocity along the longitudinal axis, in a given instant t, is defined as

$$\mathcal{E}_{\dot{x}}(t) = \dot{x}_d(t) - \dot{x}(t), \tag{3.6}$$

so that the derivative term can be rewritten as

$$\frac{\partial \mathcal{E}_{\dot{x}}(t)}{\partial t} = \frac{(\dot{x}_d(t) - \dot{x}(t)) - (\dot{x}_d(t-1) - \dot{x}(t-1))}{t - (t-1)}. \tag{3.7}$$

If the desired velocity is assumed constant ($\dot{x}_d(t) = \dot{x}_d(t-1)$), then the derivative term can be simplified as

$$\frac{\partial \mathcal{E}_{\dot{x}}(t)}{\partial t} = \frac{-\dot{x}(t) + \dot{x}(t-1)}{t - (t-1)} = \frac{-(\dot{x}(t) - \dot{x}(t-1))}{t - (t-1)} = -\frac{\partial \dot{x}(t)}{\partial t} = -\ddot{x}. \tag{3.8}$$

The same simplification can be performed for the derivative terms of the lateral and vertical

Algorithm 3.1 Auto-adjustment of hovering thrust.

procedure ADJUST_HOVERING_THRUST($\mathcal{T}_h$, $\mathcal{T}_{h\_incr}$)
  $\mathcal{T}_h$: Initial hovering thrust
  $\mathcal{T}_{h\_incr}$: Maximum hovering thrust update allowed
  $\Delta\mathcal{T}_h$ is initialized to infinity ▷ Update initialization
  while $\Delta\mathcal{T}_h$ is large **do**
    Compute the *mean* of $\mathcal{T}_d$ for some time, as long as the desired velocities are zero and the vehicle is flying high enough
    $\Delta\mathcal{T}_h = mean - \mathcal{T}_h$ ▷ Compute the new update
    if $\Delta\mathcal{T}_h > \mathcal{T}_{h\_incr}$ **then**
      $\Delta\mathcal{T}_h = \mathcal{T}_{h\_incr}$
    else if $\Delta\mathcal{T}_h < -\mathcal{T}_{h\_incr}$ **then**
      $\Delta\mathcal{T}_h = -\mathcal{T}_{h\_incr}$
    end if
    $\mathcal{T}_h = \mathcal{T}_h + \Delta\mathcal{T}_h$ ▷ The hovering thrust is updated
  end while
end procedure

velocity controllers. Thus, the expressions for the three PID controllers turn out to be

$$\theta_d(t) = K_p^{\dot{x}}\,\mathcal{E}_{\dot{x}}(t) - K_d^{\dot{x}}\,\ddot{x} + K_i^{\dot{x}}\int_0^t \mathcal{E}_{\dot{x}}(\tau)\,d\tau, \tag{3.9}$$

$$\varphi_d(t) = K_p^{\dot{y}}\,\mathcal{E}_{\dot{y}}(t) - K_d^{\dot{y}}\,\ddot{y} + K_i^{\dot{y}}\int_0^t \mathcal{E}_{\dot{y}}(\tau)\,d\tau, \tag{3.10}$$

$$\mathcal{T}_d^*(t) = K_p^{\dot{z}}\,\mathcal{E}_{\dot{z}}(t) - K_d^{\dot{z}}\,\ddot{z} + K_i^{\dot{z}}\int_0^t \mathcal{E}_{\dot{z}}(\tau)\,d\tau, \tag{3.11}$$

i.e. linear accelerations estimated by the IMU are introduced in the derivative term.

Regarding the height controller, it is activated when the desired vertical velocity $\dot{z}_d$ is zero. When this command becomes null, the platform height is saved as the desired height z_d, and used to compute the height error $\mathcal{E}_z$. A PID has been also used to implement this controller. The output of this PID is the desired vertical speed $\dot{z}_d^*$. The same derivation performed for the velocity controllers is used for the height PID controller, and the following expression is obtained:

$$\dot{z}_d^*(t) = K_p^z\,\mathcal{E}_z(t) - K_d^z\,\dot{z} + K_i^z\int_0^t \mathcal{E}_z(\tau)\,d\tau. \tag{3.12}$$

Notice that, this time, the derivative term makes use of the estimated velocity in the vertical axis. Before being introduced in the vertical speed controller, the output of this PID is saturated by means of

$$\dot{z}_d(t) = \max(-\dot{z}_{d_M},\, \min(\dot{z}_{d_M},\, \dot{z}_d^*(t))), \tag{3.13}$$

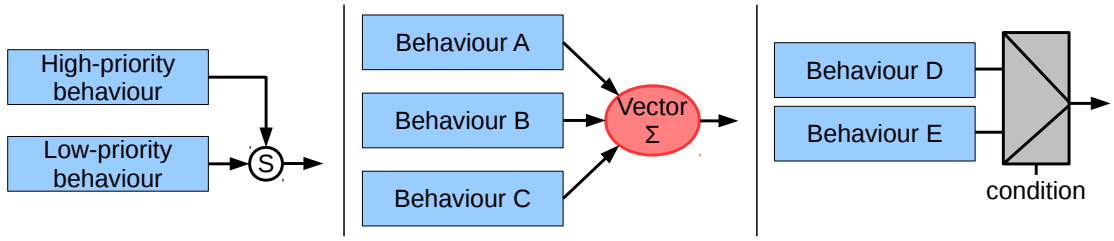


Figure 3.10: Behaviour combination mechanisms: (left) competitive mechanism using the subsumption architectural model for suppression, (middle) cooperative mechanism using motor schema for vector summation, and (right) selective mechanism by signals multiplexing.

where $\dot{z}_{d_M}$ is the maximum vertical velocity allowed. This is performed to limit the ascending/descending speed of the platform when the latter is trying to keep a certain height, as well as to reduce the effect produced by possible errors in the height estimation.

3.4.4 Behaviour-based Control

The high-level control layer executes the MAV behaviours module. Following the SA paradigm, this module comprises a set of robotic behaviours which are in charge of fulfilling the commanded task, indicated by the user/surveyor via *qualitative instructions*, while performing *self-preservation* tasks such as obstacle detection and collision avoidance. In other words, this module combines the user desired speed with the available sensor data through a reactive control strategy to provide the desired velocity command ($\dot{x}_d$, $\dot{y}_d$, $\dot{z}_d$).

The robot behaviours are organized in a hybrid competitive-cooperative framework. This framework makes use of the following combination mechanisms:

- a competitive mechanism to allow a higher priority behaviour to overwrite the output of a lower priority behaviour, which consists in using a suppression mechanism taken from the *subsumption* architectural model [178] (see Fig. 3.10 [left]);
- a cooperative mechanism to merge the output of several behaviours with the same priority level, which is performed through a *motor schema* [178], where all the behaviours involved supply each a motion vector, so that the final output is the weighted summation of all motion vectors (see Fig. 3.10 [middle]); and
- a selective mechanism to choose between the ouptput of two or more behaviours, i.e. a sort of multiplexer (see Fig. 3.10 [right]).

Figure 3.11 details our behaviour-based architecture, showing how the different behaviours are organized and how they contribute to the final speed command. The different behaviours are grouped depending on its purpose, setting up four general categories:

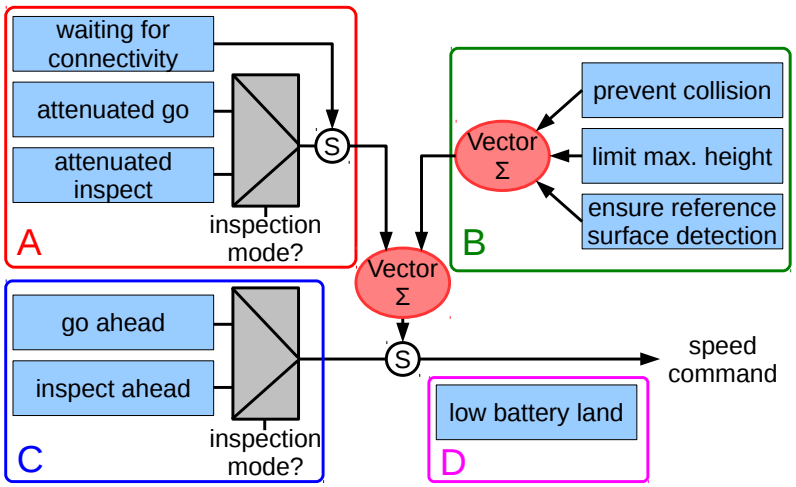


Figure 3.11: MAV behaviours: (A) groups behaviours to accomplish the user intention, (B) groups behaviours that ensure the platform safety within the environment, (C) groups behaviours that increase the autonomy level, and (D) groups behaviours oriented to check flight viability.

- *Behaviours to accomplish the user intention.* This group comprises the *attenuated_go*, the *attenuated_inspect* and the *waiting_for_connectivity* behaviours. The behaviour *attenuated_go* propagates the user desired speed vector command, attenuating it towards zero in the presence of close obstacles. In more detail, when the vehicle is moving towards an obstacle, the speed is reduced in accordance to the proximity to the obstacle. The speed is not attenuated when the user command moves the MAV away from the obstacle. By way of example, when the vehicle moves along the longitudinal axis obeying a user command $\dot{x}_{ud}$, the output of the *attenuated_go* behaviour $\dot{x}_{d\_ag}$ is computed as

$$\dot{x}_{d\_ag} = \min(\dot{x}_{ud},\, K_{ag}\, \dot{x}_{d_M} \cdot \max(0,\, d_f - d_m)), \tag{3.14}$$

  where $K_{ag} \in [0, 1]$ is the attenuation factor, $\dot{x}_{d_M}$ is the maximum speed allowed along the X axis, d_f is the estimated distance to the nearest obstacle in front of the MAV and d_m is the minimum distance allowed to any obstacle. Notice that Eq. 3.14 limits the final speed command to the user desired speed, which in turn is also limited through the user interface.

  The *attenuated_inspect* behaviour proceeds in the same way, but it is only activated in the so-called *inspection mode*. While in this mode, the vehicle moves at a constant and reduced speed (if it is not hovering) and user commands for longitudinal displacements or turning around the vertical axis are ignored. Furthermore, a PID controller, similar

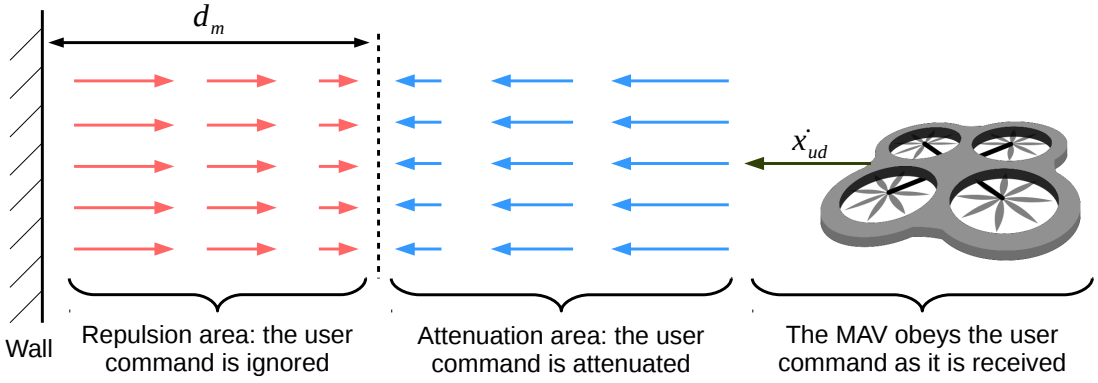


Figure 3.12: Collision avoidance functionality for the MAV approaching a wall. This results from the joint actuation of the *attenuated_go/inspect* and the *prevent_collision* behaviours.

to that used for height control, is activated to provide the suitable velocity commands along the longitudinal axis ($\dot{x}_{d\_ai}$). In this way, during an inspection, the platform keeps at a constant distance and orientation with regard to the front wall, for improved image capture.

Finally, the *waiting_for_connectivity* behaviour sets zero speed (i.e. hovering) when the connection with the base station is lost. After some seconds, if the connection is not restored, this behaviour is also in charge of landing the platform.

- *Behaviours to ensure the platform safety within the environment.* This category includes the *prevent_collision* behaviour, which generates a repulsive vector to separate the platform from surrounding obstacles, whose magnitude increases as a function of proximity. By way of example, when an obstacle is detected in front of the platform, at a distance lower than the minimum allowed (d_m), the *prevent_collision* behaviour gives rise to the following output

$$\dot{x}_{d\_pc} = -K_{pc}\,\dot{x}_{d_M} \cdot \max(0,\, d_m - d_f), \tag{3.15}$$

where $K_{pc} \in [0, 1]$ is the repulsion factor. The joint actuation of this behaviour and the *attenuated_go/inspect* behaviours implements the collision avoidance functionality on-board the platform. Figure 3.12 illustrates this joint actuation for the case of the MAV approaching a wall.

A second behaviour called *limit_max_height* produces an attraction vector towards the ground when the vehicle is approaching its maximum flight height. This is computed as

$$\dot{z}_{d\_lmh} = -K_{lmh}\,\dot{z}_{d_M} \cdot \max(0,\, z - z_M), \tag{3.16}$$

where $K_{lmh} \in [0, 1]$ is the attraction factor, and $\dot{z}_{d_M}$ and z_M are the maximum allowed values for the vertical speed and height.

A last behaviour called *ensure_reference_surface_detection* generates suitable atraction vectors that keep the platform close enough to at least one of the reference surfaces (the ground or the front wall), to ensure proper state estimations when using the optical flow sensors (see Section 3.5.1 for the details). Thus, if the vehicle requires the ground to estimate its estate (i.e. there is no wall in front of the MAV), an attraction vector towards this surface is applied when the distance d_b exceeds a maximum value d_{b_M}. The attraction vector is computed as

$$\dot{z}_{d\_ers} = -K_{ers}\, \dot{z}_{d_M} \cdot \max(0,\, d_b - d_{b_M}), \tag{3.17}$$

where $K_{ers} \in [0, 1]$ is the attraction factor. Similarly, when the vehicle requires the front wall to estimate its estate (i.e. the ground is not detected by the bottom looking optical flow sensor), an attraction vector towards the inspected wall is applied. This is computed as

$$\dot{x}_{d\_ers} = K_{ers}\, \dot{x}_{d_M} \cdot \max(0,\, d_f - d_{f_M}), \tag{3.18}$$

where d_{f_M} is the maximum distance allowed regarding the inspected wall. Furthermore, in this situation, the commands for turning around the vertical axis are suppressed to keep detecting the wall in front of the platform. This is performed through a desired angular velocity that compensates the user rotation command $\dot{\psi}_{ud}$:

$$\dot{\psi}_{d\_ers} = -\dot{\psi}_{ud}. \tag{3.19}$$

- *Behaviours to increase the autonomy level.* This category comprises the behaviours that provide higher levels of autonomy to both simplify the vehicle operation and to introduce further assistance during inspections. The *go_ahead* behaviour is in charge of keeping the user speed command, i.e. the user does not need to reiterate the command all the time, until some obstacle is detected or a new desired speed is introduced by the user. This behaviour is of special interest when a large displacement has to be performed, for example, to go to the next wall to be inspected. An analogous behaviour called *inspect_ahead* is in use when the platform is flying in *inspection mode*. This is useful, for example, when inspecting a large wall. Notice that the output of the behaviours in this category can be overwritten at any time by the behaviours in the previous mentioned categories.

- *Behaviours to check flight viability.* This group does not contribute to the speed command, but it is in charge of ensuring that the flight can start or progress at a certain

moment in time. This group includes only one behaviour named *low_battery_land* that makes the vehicle descend and land when the battery is almost exhausted, i.e. its voltage is below a minimum value v_m.

As a final note, it must be taken into account that this set of behaviours has been designed having in mind the visual inspection application.

3.5 State Estimation

The estimation of the state is performed processing and combining the data provided by the different sensors. In order to allow the different sensor suites defined in Section 3.3, the state estimation unit has been designed as a pipeline comprising several components which perform a specific task each. These components can be added or removed depending on the processes that need to be performed to get an estimate for one or more state variables from the data provided by a specific sensor. The design of the pipeline has been formalized through the following types of components:

- *Driver*. This component is used to communicate with a sensor device and to introduce the raw data into the pipeline.
- *Data filtering*. It provides different kinds of filters to block, restrict and/or smooth data. Some examples are the mean filter, the median filter and different versions of the Kalman Filter.
- *Data preparation*. This component eliminates/compensates undesired effects (such as biases or offsets) from the measured data. For example, a data preparation component can be used to eliminate the gravity acceleration from the linear acceleration measure captured with an IMU. Another component can be used to compensate the tilt (roll and pitch) of the MAV in order to obtain the distance to the ground from a bottom-looking range sensor data.
- *Data splitting*. This component allows splitting the information included in a data structure, so that several output structures are provided with a part of the information each. For example, a data splitting module can be used to divide an image into two sub-images, or to separate the different channels of a colour image.
- *Data processing*. This is the key component in charge of processing data to obtain useful information for the state estimation. This component can be used to implement processes such as odometers, SLAM methods, etc.
- *Data combination*. It is used to merge data from two or more inputs, usually provided by data processing components, to obtain a state estimation. This can entail some

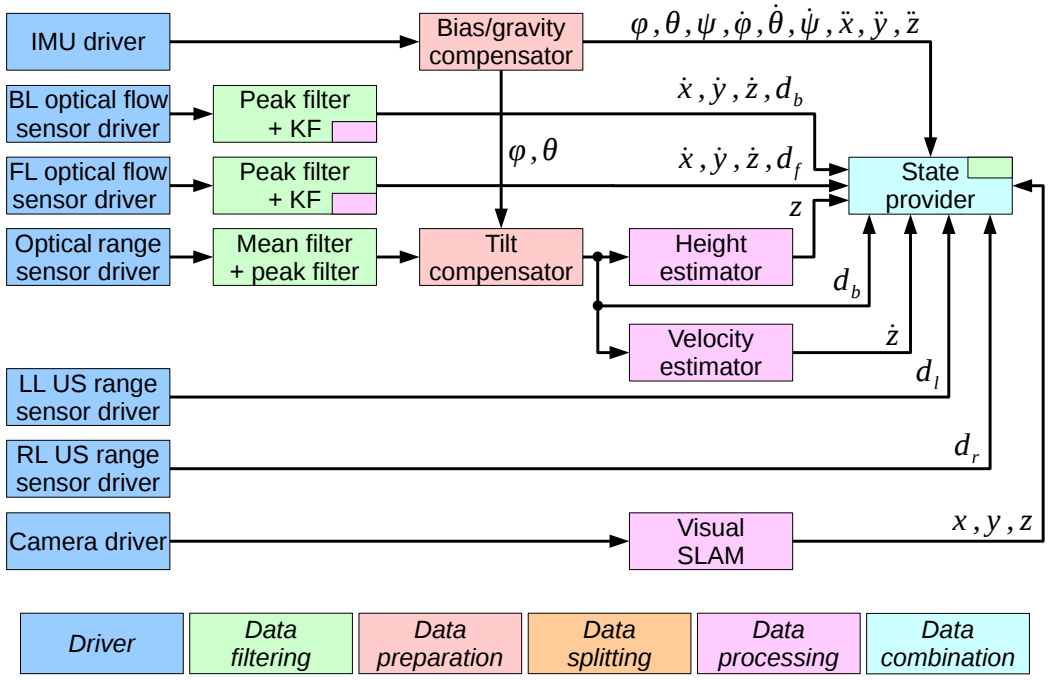


Figure 3.13: State estimation pipeline for the SS1. Sensor orientation: BL/bottom-looking, FL/forward-looking, LL/left-looking, RL/right-looking.

kind of process to select the suitable data among the input elements, according to some conditions.

Using these components, a state estimation pipeline has been defined for each one of the three sensor suites. These are detailed in the following subsections.

3.5.1 Sensor Suite 1

The state estimation pipeline designed for the SS1 is shown in Fig. 3.13. Seven *driver* components are used, one for each sensor: the IMU, two optical flow sensors, one optical range sensor, two US range sensors and one colour camera.

The *driver* component for the IMU is assumed to provide the orientation of the platform (roll, pitch and yaw), the linear accelerations and the angular velocities. All these values are usually filtered on-board the IMU device, so that further filtering is not required. Nevertheless, the measured linear accelerations are affected by the gravity acceleration (9.8 m/s^2), so that the raw values for $[\ddot{x}, \ddot{y}, \ddot{z}]$ when the vehicle is completely horizontal are [0.0, 0.0, 9.8]. To compensate this effect, a *data preparation* component is used, taking into account the platform roll and pitch, to compensate the corresponding value to each linear acceleration component.

Moreover, the linear acceleration measures are typically affected by some static bias. This is compensated in the same *data preparation* component, estimating the bias as the mean

value of the first N measures, one for each linear acceleration, and subtracting them from the gravity-compensated measures.

The complete compensation is performed using the expressions

$$\ddot{x} = \ddot{x}^* + 9.8\,\sin(\theta) - b_{\ddot{x}}, \tag{3.20}$$

$$\ddot{y} = \ddot{y}^* - 9.8\,\cos(\theta)\,\sin(\varphi) - b_{\ddot{y}}, \tag{3.21}$$

$$\ddot{z} = \ddot{z}^* - 9.8\,\cos(\theta)\,\cos(\varphi) - b_{\ddot{z}}, \tag{3.22}$$

where $\ddot{x}^*$, $\ddot{y}^*$ and $\ddot{z}^*$ are the linear accelerations provided by the IMU, before any compensation, and $b_{\ddot{x}}$, $b_{\ddot{y}}$ and $b_{\ddot{z}}$ are the acceleration biases.

The *drivers* for the optical flow sensors provide both the velocity regarding the reference surface (the ground for the bottom-looking sensor and the front wall for the forward-looking sensor) and the distance to that surface (measured by means of the embedded US range sensor). A *data filtering* component is used to filter out the peaks (if any) in the measured distance, as well as to smooth the velocities.

Firstly, a so-called peak filter is used to detect large changes in the measured distance. When this occurs, the last distance used is employed until detecting the end of the peak (i.e. the current distance becomes similar to the last used). If the peak has not finished after some time, it may indicate that this is due to a discontinuity in the reference surface (indeed it was not a peak), and the new measured distance is used. Notice that, while in normal operation, this filter does not introduce any delay into the distance measurement. The peak filter operation is described in Alg. 3.2.

Secondly, a Kalman Filter (KF) [179] is used to smooth the velocities measured with the optical flow and also to estimate the normal speed with regard to the reference surface. Notice that, in SS1, the bottom-looking optical flow sensor provides the longitudinal and lateral velocities ($\dot{x}$ and $\dot{y}$), while the vertical velocity ($\dot{z}$) is estimated from the measured distance by the KF. In the case of the forward-looking optical flow sensor, it provides the lateral and vertical velocities ($\dot{y}$ and $\dot{z}$ in the body fixed coordinate frame), and the KF is used to estimate the longitudinal speed ($\dot{x}$). Due to the estimation of these state variables via software, not through a sensing device, the corresponding components in the pipeline can be also considered as *data processing*, as indicated in Fig. 3.13.

Within the KF, the state at time k is evolved from the state at $k-1$ according to

$$\mathbf{x}_k = \mathbf{F}_k\,\mathbf{x}_{k-1} + \mathbf{w}_k, \tag{3.23}$$

where $\mathbf{F}_k$ is the state transition model which is applied to the previous state $\mathbf{x}_{k-1}$ and $\mathbf{w}_k$ is the process noise which is assumed to be drawn from a zero mean multivariate normal

Algorithm 3.2 Peak filter procedure for distance measures.

```
1: procedure PEAK_FILTER(r)
2:     r: Distance provided by a range sensor
3:     if first time then
4:         last_r = r
5:         return r
6:     end if
7:     if r is similar to last_r then                    ▷ No peak detected
8:         last_r = r
9:         return r
10:    else
11:        if last_r used in many consecutive cycles then    ▷ Discontinuity in the surface
12:            last_r = r
13:            return r
14:        else                                              ▷ Peak detected
15:            return last_r
16:        end if
17:    end if
18: end procedure
```

distribution with covariance $\mathbf{Q}_k$. In the case of the bottom-looking optical flow sensor, the state is defined as

$$\mathbf{x}_k = \begin{bmatrix} d_b \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}, \tag{3.24}$$

the state transition model $\mathbf{F}_k$ is defined as

$$\mathbf{F}_k = \begin{bmatrix} 1 & 0 & 0 & \Delta t \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.25}$$

where Δt is the elapsed time between consecutive cycles, and the process noise covariance is defined as

$$\mathbf{Q}_k = \begin{bmatrix} \frac{1}{4}\,\Delta t^4\,\sigma_z^2 & 0 & 0 & \frac{1}{2}\,\Delta t^3\,\sigma_z^2 \\ 0 & \Delta t^2\,\sigma_{xy}^2 & 0 & 0 \\ 0 & 0 & \Delta t^2\,\sigma_{xy}^2 & 0 \\ \frac{1}{2}\,\Delta t^3\,\sigma_z^2 & 0 & 0 & \Delta t^2\,\sigma_z^2 \end{bmatrix}. \tag{3.26}$$

Notice that this model corresponds to uniformly accelerated motion, where the acceleration term is introduced as the process noise by means of the covariances σ_{xy} and σ_z. In other

words, σ_{xy} models the uniform acceleration of the MAV along the X and Y axes, which can be considered coincident in a multirotor MAV, while σ_z models the acceleration along the Z axis.

For the forward-looking sensor, the expressions for the state $\mathbf{x}_k$ and the process noise covariance $\mathbf{Q}_k$ needs to be redefined as

$$\mathbf{x}_k = \begin{bmatrix} d_f \\ \dot{z} \\ \dot{y} \\ \dot{x} \end{bmatrix}, \tag{3.27}$$

$$\mathbf{Q}_k = \begin{bmatrix} \frac{1}{4}\Delta t^4 \sigma_{xy}^2 & 0 & 0 & \frac{1}{2}\Delta t^3 \sigma_{xy}^2 \\ 0 & \Delta t^2 \sigma_z^2 & 0 & 0 \\ 0 & 0 & \Delta t^2 \sigma_{xy}^2 & 0 \\ \frac{1}{2}\Delta t^3 \sigma_{xy}^2 & 0 & 0 & \Delta t^2 \sigma_{xy}^2 \end{bmatrix}. \tag{3.28}$$

At time k, a measurement $\mathbf{z}_k$ of the state $\mathbf{x}_k$ is made according to

$$\mathbf{z}_k = \mathbf{H}_k \, \mathbf{x}_k + \mathbf{v}_k, \tag{3.29}$$

where $\mathbf{H}_k$ is the observation model and $\mathbf{v}_k$ is the observation noise, which is assumed to be zero mean Gaussian noise with covariance $\mathbf{R}_k$. The measure provided by the bottom-looking sensor is

$$\mathbf{z}_k = \begin{bmatrix} d_b \\ \dot{x} \\ \dot{y} \end{bmatrix}, \tag{3.30}$$

while for the forward looking sensor is

$$\mathbf{z}_k = \begin{bmatrix} d_f \\ \dot{z} \\ \dot{y} \end{bmatrix}. \tag{3.31}$$

The matrices $\mathbf{H}_k$ and $\mathbf{R}_k$ for both sensors are defined as

$$\mathbf{H}_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{3.32}$$

$$\mathbf{R}_k = \begin{bmatrix} \sigma_{dist} & 0 & 0 \\ 0 & \sigma_{vel'} & 0 \\ 0 & 0 & \sigma_{vel''} \end{bmatrix}, \tag{3.33}$$

Algorithm 3.3 Height estimation procedure.

```
1: procedure HEIGHT_ESTIMATOR(d)
2:     d: Distance to the floor
3:     if first time then
4:         last_d = d
5:         MAV_height = 0
6:         floor_height = 0
7:         return MAV_height
8:     end if
9:     Δd = d − last_d
10:    if Δd is small enough then          ▷ Change in the estimated MAV height
11:        MAV_height = MAV_height + Δd
12:    else                                ▷ Discontinuity in the floor
13:        floor_height = floor_height − Δd
14:    end if
15:    last_d = d
16:    return MAV_height
17: end procedure
```

where σ_{dist}, $\sigma_{vel'}$ and $\sigma_{vel''}$ are the covariances for the distance and the two velocity measures.

The optical range sensor is added to measure the distance to the floor with a detection range larger than the provided by the US sensor embedded in the optical flow device. The data provided by the corresponding driver is introduced in a third *data filtering* component. This makes use of a mean filter to remove the high-frequency noise, and a peak filter, as used for the distance provided by the optical flow sensors. The resulting distance is introduced in a *data preparation* component which compensates the MAV tilt, to obtain the distance to the floor, by means of

$$d_b = d_b^* \cos(\varphi) \cos(\theta), \tag{3.34}$$

where d_b^* is the raw distance. The resulting measure is introduced in two *data processing* components. On the one hand, the MAV height (z) is estimated in a so-called height estimator. This component keeps the values for the estimated heights of the platform and the floor, both initialized to zero. When a new distance measure d_b is received, this component computes the difference Δd regarding the previous distance received. If this value is below a certain threshold, it is considered a change in the flight height, and Δd is added to the estimated MAV height. If Δd is above the threshold, the distance change is probably due to a discontinuity in the floor, so that Δd is subtracted from the estimated floor height, while the MAV height is preserved. Notice that both heights are always referenced to the take-off surface. The height estimation procedure is detailed in Alg. 3.3.

On the other hand, the tilt-compensated distance is also used to estimate the vertical speed $\dot{z}$. The corresponding component computes the instantaneous velocity by means of differentiation. If the result is above a given threshold, probably due to a discontinuity in

Algorithm 3.4 Vertical speed estimation procedure.

```
1: procedure VELOCITY_ESTIMATOR(d)
2:     d: Distance to the floor
3:     if first time then
4:         last_d = d
5:         return 0
6:     end if
7:     speed = (d − last_d)/Δt
8:     if speed is too high then                ▷ Discontinuity in the floor
9:         speed = 0
10:    end if
11:    Smooth speed using a KF
12:    last_d = d
13:    return speed
14: end procedure
```

the floor surface or due to an error in the distance sensor, the velocity measure is considered incorrect and it is set to zero. The filtered velocity is finally introduced in a KF to smooth its value, as performed for the velocities provided by the optical flow sensors. The entire procedure is described in Alg. 3.4.

The drivers for the side-looking US range sensors provide the distance to the obstacles situated to the left (d_l) and to the right (d_r) of the platform. This values do not require any filtering nor preparation.

Finally, the camera driver provides colour images from the environment situated in front of the MAV. These images are introduced into a *data processing* component which performs a SLAM process. This method provides the position and orientation of the platform regarding the take-off location (further details are provided in Section 3.6.2). Since this is based on a monocular camera, the resulting positions must be scaled using a λ factor, as explained below.

All the estimated state variables are introduced in a *data combination* component, the state provider. This component combines all the estimated state variables to build up the MAV state. The values for x and y are taken from the monocular SLAM algorithm and scaled using a λ factor. This is computed dividing the estimated height z, which is taken from the height estimator, by the scaled z provided by the SLAM method. The orientation (φ, θ, ψ), angular velocities ($\dot{\varphi}$, $\dot{\theta}$, $\dot{\psi}$), linear accelerations ($\ddot{x}$, $\ddot{y}$, $\ddot{z}$), and the distances d_f, d_l and d_r, are taken from their unique providers, as shown in Fig. 3.13. The distance d_b is taken from the optical range sensor, while the one provided by the bottom looking optical flow sensor is just used to know whether this sensor is detecting the reference surface, as explained below.

Finally, the linear velocities ($\dot{x}$, $\dot{y}$, $\dot{z}$) result from combining the information provided by the bottom-looking and forward-looking optical flow sensors, as well as by the optical range sensor (just information for $\dot{z}$). The suitable source is selected in every case depending on whether the reference surfaces are detected or not. The rules for this selection are detailed in

Table 3.3: Selection of the source for the MAV velocities and height state variables, when using the SS1.

Mode	**Sensor availability**			**Sensor selection**			
	BL optF	**FL optF**	**optR**	z	$\dot{x}$	$\dot{y}$	$\dot{z}$
0	OK	N/A	OK	optR	BL optF	BL optF	optR*
1	OK	OK	OK	optR	BL optF	BL optF	FL optF
2	N/A	OK	OK	optR	FL optF*	FL optF	FL optF
3	N/A	OK	N/A	$\int \dot{z}$	FL optF*	FL optF	FL optF
-1	N/A	N/A	OK	optR	0	0	optR*
-2	N/A	N/A	N/A	$\int val'$	0	0	val''

BL optF refers to the bottom-looking optical flow sensor.
FL optF refers to the forward-looking optical flow sensor.
optR refers to the optical range sensor.
OK means that the sensor data is available.
N/A means that the sensor data is not available.
* indicates a derivation of a range measurement.
val' and val'' are positive values.

Table 3.3.

Four different modes (modes 0 to 3) are defined depending on whether the front wall and/or the ground can be used as reference surface by the optical flow sensors (i.e. whether they are closer than the maximum detection range of the embedded US range sensor). Following this selection rules, the MAV velocities are preferably estimated based on optical flow measures, and only when these are not available, the system makes use of the values obtained differentiating distance measures. Notice that the flight height is not limited as long as there is a wall in front of the vehicle that can be used as reference surface by the forward looking sensor (mode 3 is used in that case). The detection of at least one reference surface (i.e the front wall or the ground) is guaranteed thanks to the *ensure_reference_surface_detection* behaviour (see Section 3.4.4). Nevertheless, in case this behaviour can not manage to achieve its goal, two additional error modes are defined. The first one (mode -1) is used when the optical range sensor is able to detect the ground, so that this is used to estimate both the height and the vertical velocity. The second error mode (mode -2) is used when no sensor can detect any reference surface. In that case, the height value is increased using a predefined ramp, while the vertical speed is set to a fixed positive value. This error mode has been designed to make the platform descend in such an emergency situation (the PID controllers for height and vertical speed will reduce the motor thrust trying to decrease the positive ascending speed).

The selected linear velocities are finally filtered in a KF, which is implemented in the same *data combination* component (see Fig. 3.13). This is used to combine the estimated linear velocities with the linear accelerations provided by the IMU. Within this filter, the state, the

transition model and the process noise covariance are defined as

$$\mathbf{x}_k = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}, \tag{3.35}$$

$$\mathbf{F}_k = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.36}$$

$$\mathbf{Q}_k = \begin{bmatrix} \frac{1}{4}\Delta t^4 \sigma_{xy}^2 & 0 & 0 & \frac{1}{2}\Delta t^3 \sigma_{xy}^2 & 0 & 0 \\ 0 & \frac{1}{4}\Delta t^4 \sigma_{xy}^2 & 0 & 0 & \frac{1}{2}\Delta t^3 \sigma_{xy}^2 & 0 \\ 0 & 0 & \frac{1}{4}\Delta t^4 \sigma_z^2 & 0 & 0 & \frac{1}{2}\Delta t^3 \sigma_z^2 \\ \frac{1}{2}\Delta t^3 \sigma_{xy}^2 & 0 & 0 & \Delta t^2 \sigma_{xy}^2 & 0 & 0 \\ 0 & \frac{1}{2}\Delta t^3 \sigma_{xy}^2 & 0 & 0 & \Delta t^2 \sigma_{xy}^2 & 0 \\ 0 & 0 & \frac{1}{2}\Delta t^3 \sigma_z^2 & 0 & 0 & \Delta t^2 \sigma_z^2 \end{bmatrix}, \tag{3.37}$$

where σ_{xy} and σ_z model the rate of change of the accelerations, i.e the jerk, of the MAV along the different axes.

The update of the KF state is performed when the linear velocities or accelerations are received. The measurement, the observation model and the observation noise covariance for the updates using the linear velocities are defined as

$$\mathbf{z}_k = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}, \tag{3.38}$$

$$\mathbf{H}_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \tag{3.39}$$

$$\mathbf{R}_k = \begin{bmatrix} \sigma_{vel\_xy} & 0 & 0 \\ 0 & \sigma_{vel\_xy} & 0 \\ 0 & 0 & \sigma_{vel\_z} \end{bmatrix}, \tag{3.40}$$

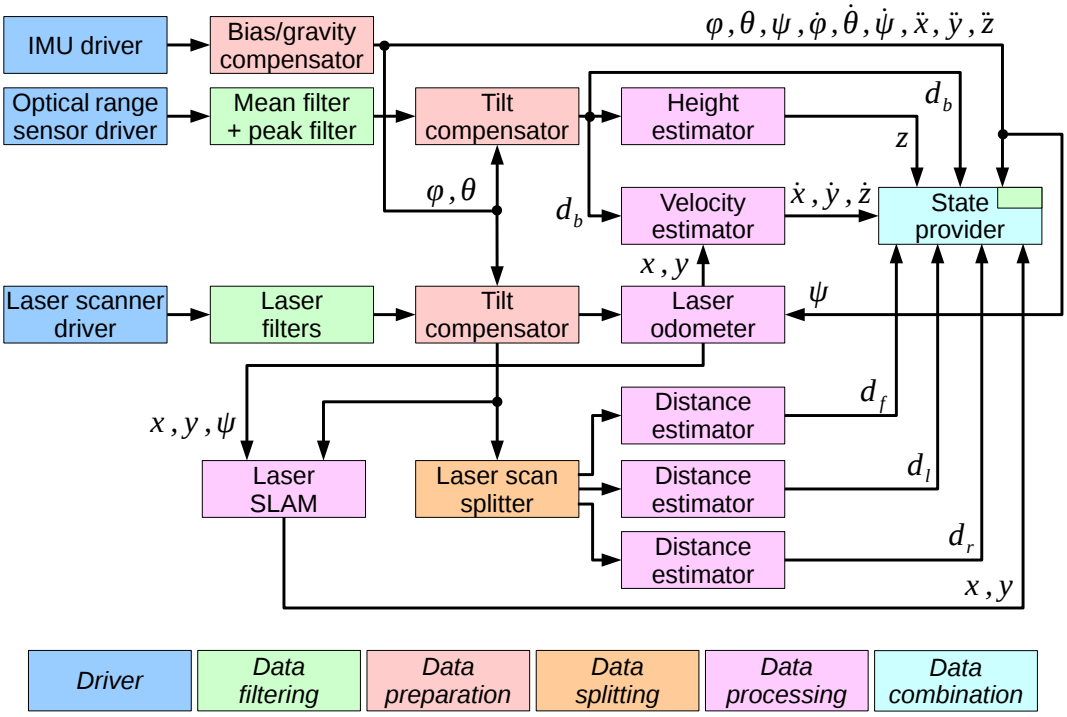


Figure 3.14: State estimation pipeline for the SS2.

while, for the updates using the linear accelerations, these are defined as

$$\mathbf{z}_k = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}, \tag{3.41}$$

$$\mathbf{H}_k = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.42}$$

$$\mathbf{R}_k = \begin{bmatrix} \sigma_{acc\_xy} & 0 & 0 \\ 0 & \sigma_{acc\_xy} & 0 \\ 0 & 0 & \sigma_{acc\_z} \end{bmatrix}. \tag{3.43}$$

3.5.2 Sensor Suite 2

Figure 3.14 shows the state estimation pipeline for the case of SS2. The IMU and the optical range sensor are used to estimate the same estate variables as for the SS1 pipeline, so that the same components are used.

The laser scanner driver provides the distances to the obstacles situated around the sensor, giving a measurement every α_{incr} degrees, from a minimum angle α_{min} to a maximum angle

α_{max}. This array of distances, i.e. a laser scan, is introduced in a *data filtering* component which applies two filters. On the one hand, a filter is used to remove laser readings that are most likely caused by the veiling effect, which is produced when the edge of an object is being scanned. For any two points p_1 and p_2 corresponding to the measures of two consecutive beams in the laser scan, the perpendicular angle $\angle Op_1p_2$ is computed, where O is the origin of the laser. If the perpendicular angle is less than a minimum value or greater than a maximum value, all the neighbours further away than those measures are removed.

On the other hand, we apply a range filter to remove all measurements which are greater than an upper value or less than a lower value. This filter allows removing, for example, all laser beams which collide with some element of the MAV structure.

The filtered laser scan is introduced in a *data preparation* component which compensates the roll and pitch of the MAV, in order to obtain the orthogonal projection of the laser scan. Firstly, the laser scan is converted to a 3D point cloud, where each beam b_i of the laser scan results into a point p_i with coordinates $[x_i, y_i, z_i]$ computed as

$$\begin{cases} x_i = d_i \cos\alpha_i \\ y_i = d_i \sin\alpha_i \\ z_i = 0 \end{cases}, \tag{3.44}$$

where $d_i \geq 0$ and $\alpha_i \in [\alpha_{min}, \alpha_{max}]$ are the distance and angle associated to the beam b_i. Then, the resulting point cloud $\mathcal{P}^*$ is rotated in 3D space, using the rotation matrix which compensates the platform roll and pitch:

$$\mathcal{P} = \begin{bmatrix} \cos\theta & \sin\theta\sin\varphi & \sin\theta\cos\varphi \\ 0 & \cos\varphi & -\sin\varphi \\ -\sin\theta & \cos\theta\sin\varphi & \cos\theta\cos\varphi \end{bmatrix} \mathcal{P}^*. \tag{3.45}$$

The x and y coordinates of the points in the resulting point cloud $\mathcal{P}$ represent the orthogonal projection of the laser scan (the z coordinate is unused). Finally, the point cloud is converted back to a laser scan. To do that, the distance and the angle of each point regarding the sensor is computed. This process is described in Alg. 3.5.

A laser-based odometer is used next to estimate first the 2D motion of the platform and, ultimately, it 2D location. This *data processing* component makes use of an Iterative Closest Point (ICP) algorithm to estimate the 2D transform T (comprising a translation and a rotation) necessary to match the current laser scan with the previous laser scan. Indeed, the ICP algorithm operates with point clouds, which are computed by means of Eq. 3.44. Then, to align the point cloud $\mathcal{P}$ to the reference $\mathcal{P}_{ref}$, the transform T is computed using the mean squared error cost function

$$e_k = \frac{1}{N}\sum_{i=1}^{N} \parallel q_{ik} - \mathrm{Rot}_k(p_i) - \mathrm{Trans}_k \parallel^2, \tag{3.46}$$

Algorithm 3.5 Procedure to convert a point cloud into a laser scan.

procedure POINT_CLOUD_TO_LASER_SCAN($\mathcal{P}$)
  $\mathcal{P}$: Point cloud
  Initialize all the beams of *laser_scan* to infinity
  for all the points p in $\mathcal{P}$ **do**
    $d = \sqrt{p.x^2 + p.y^2}$ ▷ Distance to the sensor
    $\alpha = \arctan(p.y/p.x)$ ▷ Angle regarding the sensor
    $b = (\alpha - \alpha_{min})/\alpha_{incr}$ ▷ Laser scan beam
    $laser\_scan(b) = d$
  end for
  return *laser_scan*
end procedure

which determines the alignment error at iteration k, where p_i refers to each point in $\mathcal{P}$, N is the number of elements in $\mathcal{P}$, q_i is the point from $\mathcal{P}_{ref}$ which is closest to p_i, and Rot_k and $Trans_k$ are the rotation and the translation included in transform T at instant k. The ICP algorithm proceeds as indicated in Alg. 3.6. Notice that, in this algorithm, the rotation in yaw (ψ) provided by the IMU is used as initial estimation of the transform (i.e. initial guess). Furthermore, the reference point cloud $\mathcal{P}_{ref}$ is kept during several executions, and it is just updated when the platform performs a large displacement. This reduces the drift in the estimated pose produced by the noise in the scans, which can lead to non-zero transforms even when there is no displacement.

The resulting 2D location (x, y) is not used as the position estimate due to the inherent bias in dead-reckoning processes, but it is introduced in a 3-axis velocity estimator together with the estimated distance to the ground (d_b). This *data processing* component is analogous to the vertical speed estimator used in in the SS1 (see Alg. 3.4), but defined for the three linear velocities. Like the one-dimensional version, this component includes a step to filter out peaks in the computed speed, and a KF to smooth the final value. This time, the state, the transition model and the process noise covariance of the KF are defined as

$$\mathbf{x}_k = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}, \tag{3.47}$$

$$\mathbf{F}_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.48}$$

$$\mathbf{Q}_k = \begin{bmatrix} \Delta t^2\, \sigma_{xy}^2 & 0 & 0 \\ 0 & \Delta t^2\, \sigma_{xy}^2 & 0 \\ 0 & 0 & \Delta t^2\, \sigma_z^2 \end{bmatrix}, \tag{3.49}$$

Algorithm 3.6 ICP algorithm to implement a laser scan odometer.

procedure LASER_ODOMETER($\mathcal{P}$, ψ)
  $\mathcal{P}$: Point cloud
  ψ: yaw
  if first time **then**
    $last_\psi = \psi$
    $\mathcal{P}_{ref} = \mathcal{P}$ ▷ Take the initial point cloud as reference
    Initialize the transforms T_{global} and T_{ref} to the identity
    return T_{global}
  end if
  $\Delta\psi = \psi - last_\psi$
  Initialize the transform T with the rotation $\Delta\psi$
  Transform $\mathcal{P}$ using T
  Compute the *alignment_error* between $\mathcal{P}$ and $\mathcal{P}_{ref}$
  while *alignment_error* is large **do**
    for all the points p in $\mathcal{P}$ **do**
      Find the closest point q in $\mathcal{P}_{ref}$
    end for
    Estimate T that best aligns each p in $\mathcal{P}$ to its match q in $\mathcal{P}_{ref}$
    Transform $\mathcal{P}$ using T
    Update the *alignment_error*
  end while
  $T_{global} = T_{ref}\, T$ ▷ Update the transform to the origin
  if T entail a large displacement **then**
    $\mathcal{P}_{ref} = \mathcal{P}$ ▷ Update the reference point cloud
    $T_{ref} = T_{global}$
  end if
  return T_{global}
end procedure

while the measure, the observation model and the observation noise covariance are defined as

$$\mathbf{z}_k = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}, \tag{3.50}$$

$$\mathbf{H}_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.51}$$

$$\mathbf{R}_k = \begin{bmatrix} \sigma_{vel\_xy} & 0 & 0 \\ 0 & \sigma_{vel\_xy} & 0 \\ 0 & 0 & \sigma_{vel\_z} \end{bmatrix}. \tag{3.52}$$

A *data splitting* component is used to split the orthogonal laser scan into three segments, where each part comprises the beams providing information of the obstacles situated to re-

Table 3.4: Selection of the optical flow data when using the SS3.

Mode	**Sensor availability**		**Sensor selection**	
	BL optF	**FL optF**	$\dot{x}$	$\dot{y}$
0	OK	N/A	BL optF	BL optF
1	OK	OK	BL optF	BL optF
2	N/A	OK	FL optF*	FL optF
-1	N/A	N/A	0	0

BL optF refers to the bottom-looking optical flow sensor.
FL optF refers to the forward-looking optical flow sensor.
OK means that the sensor data is available.
N/A means that the sensor data is not available.
* indicates a derivation of a range measurement.

spectively the left, in front and to the right of the platform. Each segment is introduced in a *data processing* component which estimates the corresponding distance d_l, d_f or d_r as the minimum value among the readings.

As in the pipeline designed for the SS1, the 2D position of the platform (x, y) is estimated in a *data processing* component which implements a SLAM process. This makes use of the tilt-compensated laser scan and the position estimated by the odometer to create a 2D map of the environment and to compute the drift-free position of the MAV. Further details are provided in Section 3.6.2.

Finally, a *data combination* component is used to collect and provide all the estimated state variables. Unlike the SS1, in this case, each state variable has a unique provider so it is not required to perform any kind of sensor selection. The same KF used for the SS1 is of application here to filter the linear velocities fused with the linear accelerations.

3.5.3 Sensor Suite 3

The pipeline for the SS3 is detailed in Fig. 3.15. This looks very similar to SS2, but including the information provided by the two optical flow sensors. This entails the use of the two *drivers* and the two corresponding *data filtering* components already used in the pipeline for SS1. The information provided by these two devices is merged into an additional *data combination* component. Within this, a selection of the suitable state variables describing the 2D velocity ($\dot{x}$, $\dot{y}$) is performed following a subset of the rules defined for the SS1. These rules are detailed in Table 3.4.

The estimated 2D velocity is introduced, together with the estimated yaw, into the laser odometer. This is the key component within this pipeline, since it fused the estimates provided by the optical flow sensors with the laser scanner data. In this case, the odometer makes use of the estimated 2D linear velocities to get an initial estimate of the displacement of the MAV. This translation, together with the rotation indicated by the IMU, is used to initialize the

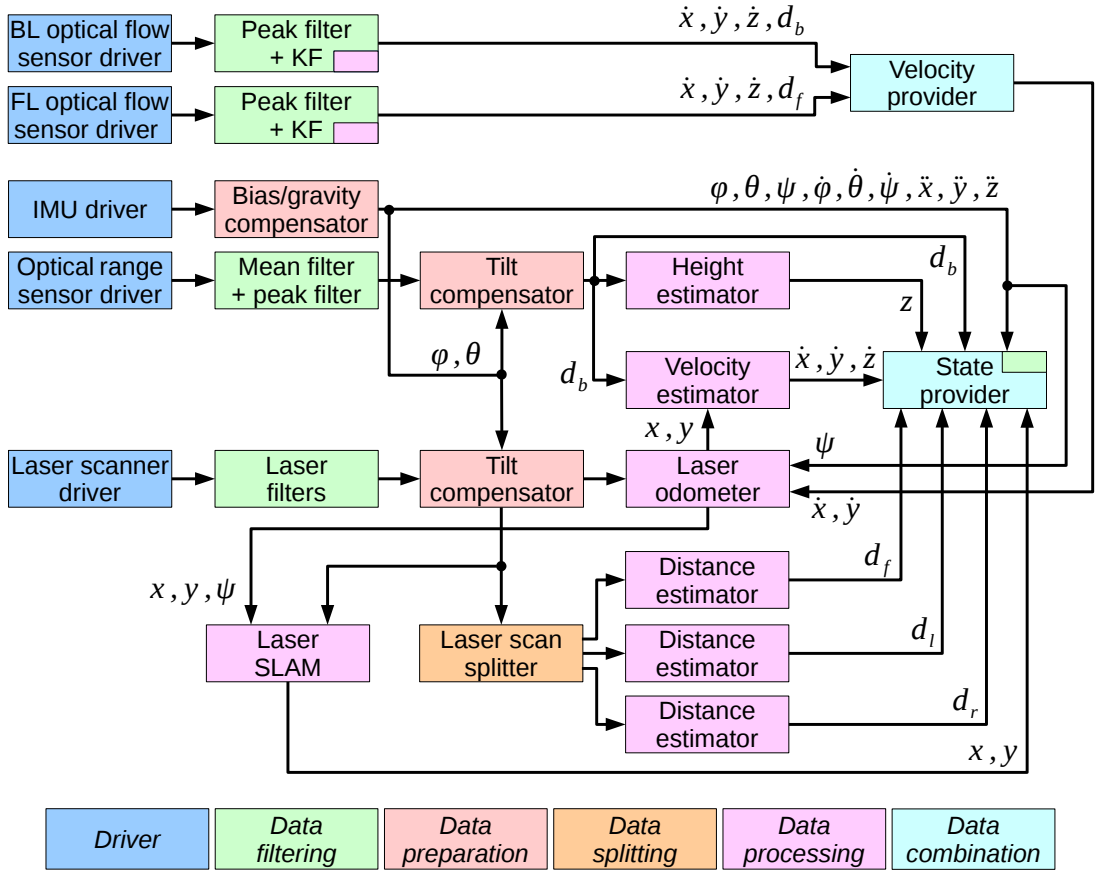


Figure 3.15: State estimation pipeline for the SS3.

ICP algorithm. In this way, when the vehicle is flying in a poorly structured environment (such as a corridor or a single large wall without corners), where the ICP algorithm fails, the displacement can be successfully estimated thanks to the optical flow measurements. The rest of the pipeline is configured in the same way as for the SS2.

3.6 Implementation

This section provides details regarding the implementation of the aerial platform. Firstly, it tackles the physical realization of the MAV, showing the specific details for the integration of the different sensor suites considered, which result in a different vehicle configuration each (Section 3.6.1). Secondly, it provides details for the implementation of the software corresponding to all the control and estate estimation systems/modules described previously in this chapter (Section 3.6.2).

Table 3.5: Features of the robotic platforms.

	Hummingbird	**Firefly**	**Pelican**
Configuration	Quadcopter	Hexacopter	Quadcopter
Size (mm)	465×465×85.5	605×665×165	590×590×188
MTOW (g)	710	1600	1650
Payload (g)	200	600	650
Flight time (min)	20	12-14	16
Max. airspeed (m/s)	15	15	16
Max. climb rate (m/s)	5	8	8
Max. thrust (N)	20	36	36

Size is given as *Length*×*Width*×*Height*.
MTOW is the maximum take-off weight according to the manufacturer.

3.6.1 Physical Realization of the Aerial Platform

For the physical implementation of our design, three different commercial multirotors produced by *Ascending Technologies*[2] (AscTec) have been used: a *Hummingbird*, a *Firefly* and a *Pelican*. These are electric-powered platforms available in our laboratory, that fulfil the requirements regarding the vehicle configuration, capabilities (such as VTOL), size and weight, and hence they are suitable for flying in confined spaces or close to structures. Table 3.5 details the main features of these platforms, as provided by the manufacturer.

These platforms incorporate one IMU and two ARM7 processors. The primary ARM7, which is called the Low-Level Processor (LLP), is in charge of executing the low-level control layer, comprising the attitude and thrust controllers. The LLP is also in charge of providing the inertial data from the IMU at 1 kHz. The secondary ARM7, the High-Level Processor (HLP), is left free so that the user can implement its own position/velocity controller. A serial connection is available to communicate both microcontrollers.

The Hummingbird is the smallest of the three platforms (see Fig. 3.17 [A]). This quadcopter has been used as test bench, so that all the algorithms to implement the different systems/modules within the control architecture have been firstly tested using this platform. Due to its limited payload (200 g), a laser scanner can not be carried by the Hummingbird, so that this platform can only fit the SS1. The optical flow sensors installed are the PX4Flow device developed within the PX4 Autopilot project [180]. This sensor, which is shown in Fig. 3.16 [left], comprises a CMOS high-sensitivity imaging sensor, an ARM Cortex M4 microcontroller to compute the optical flow at 250 Hz, a 3-axis gyroscope for angular rate compensation, and a MaxBotix US range sensor HRLV-EZ4, with 1 mm resolution, used to scale the optical flow to metric velocity values. Two additional HRLV-EZ4 US range sensors (see Fig. 3.16 [right]) estimate the distance to the obstacles situated at both sides of the

[2] http://www.asctec.de/en/

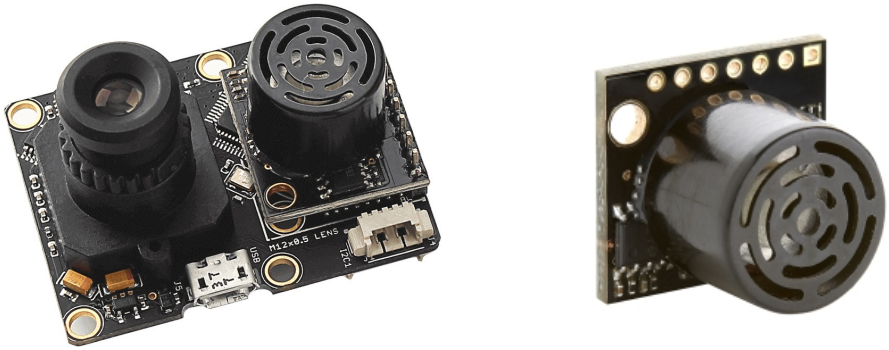


Figure 3.16: Optical flow and US range sensors used to implement the SS1: (left) PX4Flow optical flow sensor and (right) Maxbotix HRLV-MaxSonar-EZ4 US range sensor.

platform. This sensor provides measurements with 1 mm resolution at 10 Hz, weighs just 5 g and its detection range is up to 5 m. As optical range sensor, we make use of a Teraranger One [181] (see Fig. 3.3 [left]). This consists in an IR time-of-flight measurement sensor that weighs less than 20 g and can detect an obstacle situated up to 14 m far away. The camera installed is a uEye UI-1221LE (see Fig. 3.6 [left]). This is a USB 2.0 CMOS camera which provides images with a resolution of 752×480 pixels. Finally, an on-board computer has been installed to execute the high-level control, i.e. the MAV behaviours module, as well as the state estimation module. This is a Commell LP-172 Pico-ITX board featuring an Intel Atom 2×1.86 GHz processor and 4 GB RAM. Due to the limited computation resources of this board, the visual-SLAM algorithm can not be executed on-board the Hummingbird. This processing board communicates with the HLP using a serial connection. The final configuration of the Hummingbird platform is shown in Fig. 3.17 [B].

The SS1 has been also installed on-board the Firefly platform (see Fig. 3.18 [A]). This is a hexacopter with a higher payload capacity (600 g) that has been used to carry a more powerful on-board computer, which allows executing the visual-SLAM algorithm. This is an AscTec Mastermind board featuring an Intel Core 2 Duo SL9400 2×1.86 GHz processor and 4 GB RAM. Regarding the sensors, the same devices installed on the Hummingbird have been used for the Firefly. Only the optical range sensor has been changed by another device providing a larger detection range. This is a Lidar-Lite laser range finder (see Fig. 3.3 [right]) that provides range data up to 40 m at 50 Hz. Its weight, including an Arduino Nano board [3], used as an I2C to USB converter, is 37 g. Apart form the sensors necessary for the state estimation, this platform has been equipped with a GoPro Hero 4 camera to take first-person videos during the inspection mission. The complete configuration of the Firefly platform is shown in Fig. 3.18 [B].

Finally, the Pelican platform (see Fig. 3.19 [A]) is used to implement both the SS2 and the

[3]https://www.arduino.cc/en/Main/ArduinoBoardNano

Figure 3.17: The AscTec Hummingbird platform: (A) the aerial platform as delivered by the manufacturer, and (B) the platform equipped with the SS1 and an on-board computer.

SS3. The larger payload capabilities of the Pelican (650 g) together with its layered structure, allows fitting the vehicle with a laser scanner and a powerful on-board computer. Regarding the laser scanner, a Hokuyo UST 20LX has been installed (see Fig. 3.4 [right]). This is a lightweight device (only 130 g) that can detect obstacles situated up to 20 m. The optical range sensor used is the Lidar-Lite also installed on-board the Firefly platform. Regarding the on-board computer, it is an Intel NUC board featuring an Intel Core i5-4250-U 2×1.3 GHz processor and 8 GB RAM. The vision system installed on-board the Pelican includes a PointGrey Chameleon3 USB 3.0 device, which provides images up to 1288×964 pixels, and a GoPro Hero 4, as it is used on the Firefly platform. Furthermore, this platform has been fitted with a high power LED to illuminate the inspected surface in dark environments. To implement the SS3, two PX4Flow sensors are added to the previous configuration. Figure 3.19 [B] shows the Pelican platform fitted with the SS2.

Notice that the three platforms have been modified to accommodate the aforementioned equipment. In this regard, Table 3.6 details the new dimensions and weights for the three

Figure 3.18: The AscTec Firefly platform: (A) the aerial platform as delivered by the manufacturer, and (B) the platform equipped with the SS1, an on-board computer and an additional camera for video recording.

vehicles once they have been modified and equipped.

Regarding the base station, we have used a generic laptop featuring an Intel Core 2 Duo T6670 2×2.20 GHz processor and 4 GB of RAM. A joystick or gamepad is connected to this laptop to allow the user/surveyor to introduce the commands. The base station communicates with the on-board computers installed on the three MAVs via a WiFi connection. To this end, the involved machines can use both the 2.4 GHz and the 5 GHz bands.

3.6.2 Software Organization

To implement the robotic system, we have developed software to be executed on the three different processing units/boards available: the secondary ARM7 processor (HLP according to the manufacturer nomenclature), the on-board computer and the base station. The

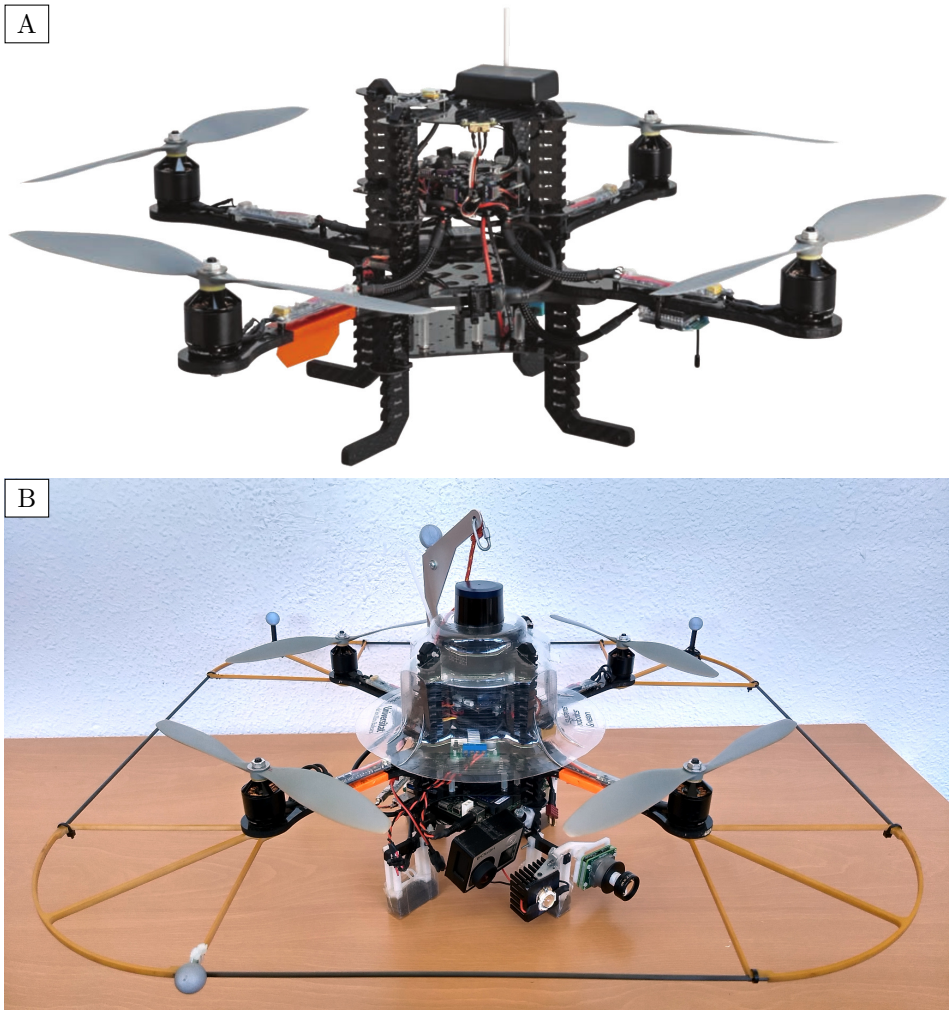


Figure 3.19: The AscTec Pelican platform: (A) the aerial platform as delivered by the manufacturer, and (B) the platform equipped with the sensor suite 2, an on-board computer, an additional camera for video recording and a high power LED.

software for the HLP includes the implementation of the flying state machine, described in Section 3.4.1, and the mid-level control layer, comprising the height and velocity controllers. The code for these controllers has been designed and generated using the Matlab Simulink [4] environment. The HLP processor has been also programmed to execute the bias/gravity compensator component, included in the state estimation pipeline, to prepare the data provided by the IMU (see Section 3.5.1 for details).

Both the on-board computer and the base station run Linux Ubuntu. The software developed for these machines has been programmed using the Robot Operating System (ROS) [5] [182]. This is a widely used open-source set of software libraries and tools for the

[4] https://www.mathworks.com/products/simulink/
[5] http://www.ros.org/

Table 3.6: Features of the three platforms equipped with the corresponding sensor suite.

	Hummingbird	**Firefly**	**Pelican**
Size (mm)	465×465×190	605×665×205	590×590×275
Weight (g)	1060	1895	1845

Size is given as *Length*×*Width*×*Height*.

development of robot applications. ROS libraries contain state-of-the-art algorithms for a large variety of purposes, several drivers for sensors and actuators, and different tools (for data visualization, simulation, debugging, etc.) which simplify the development of robot-based systems. Among its multiple features, we highlight the management of the information exchanged between different pieces of software, or ROS nodes. This is performed in a transparent manner for the user, even when the information flows between nodes executed in different machines.

Each component in the state estimation pipeline has been programmed as a ROS node (excluding the bias/gravity compensator). All these nodes, which are executed on the on-board computer, have been implemented following the specifications provided in Section 3.5. The laser odometer has been implemented adapting the code by I. Dryanovski, W. Morris and A. Censi [6], which makes use of the canonical scan matcher published in [183].

Regarding the SLAM algorithms, we have integrated two existing methods. On the one hand, the monocular version of the visual SLAM method *ORB-SLAM* [184] has been integrated in the state estimation pipeline executed on-board the AscTec Firefly platform. On the other hand, the laser-based SLAM method *GMapping* [185] has been integrated as part of the pipelines designed for the SS2 and the SS3, both using the laser scanner and executed on-board the AscTec Pelican.

The MAV behaviours module has been developed as another ROS node which comprises several functions to implement the different robot behaviours described in Section 3.4.4. This node receives the user commands and the state of the platform, and provides the final commands to be sent to the mid-level control layer, executed on the HLP. These commands are sent to an additional node which implements an interface between the HLP processor and the ROS software. In more detail, this node sends, through the serial communication with the HLP, the velocity, take-off and landing commands, while provides the other ROS nodes with the IMU data and information about the platform status: the flight stage, the linear acceleration biases, and the battery voltage.

To manage the camera during an inspection, a camera module has been implemented. This consists in a ROS node which communicates with the camera driver. This module allows taking a single picture on demand, as well as taking a sequence of images at a specified frame rate. When taking a single image, this is sent to the base station for its visualization.

[6] http://wiki.ros.org/laser_scan_matcher

Table 3.7: CPU load and memory usage for the different processing boards. Metrics for the on-board computer provided excluding/including the execution of the corresponding SLAM method.

	HLP	**On-board computer**		
		Hummingbird	**Firefly**	**Pelican**
CPU load	62%	33%/–	7%/65%	6%/12%
Memory usage	–	15%/–	15%/66%	27%/40%

Nevertheless, image sequences are stored on-board the MAV to reduce network traffic. The images in these sequences are tagged with the vehicle position, and represent the output of the inspection performed using the MAV.

Table 3.7 shows measures for the CPU load and the memory usage, regarding the different processing boards/units installed on-board the MAVs. These measures are given both excluding and including the execution of the corresponding SLAM algorithm, which is the costliest process. As can be observed, when the SLAM algorithm is not considered, the SS2 pipeline executed on the Pelican platform requires more memory than the SS1, executed on the other platforms. This is due to the the different filtering and pre-processing stages required to prepare the data provided by the laser scanner. Nevertheless, when the SLAM methods are executed, the vision-based method included in the SS1 requires more memory to store all the data structures that this algorithm handles. Notice that the percentages provided in this table are illustrative, since the memory consumption will vary depending on the size of the map and the configuration of the algorithm parameters.

Two ROS nodes run on-board the base station. The first one consists in a driver to manage the joystick/gamepad device. The second node receives the status of all the axes and buttons of the input peripheral, and provides the user commands to the different modules on the MAV:

- the user desired velocities along the three axes, and the rotational velocity around the vertical axis, are provided to the MAV behaviours module,
- the take-off/land commands are provided to the HLP interface node,
- the enable/disable *inspection mode* is provided to the MAV behaviours module,
- the command to keep the current speed (i.e. to activate the *go/inspect_ahead* behaviour) is provided to the MAV behavious module, and
- the command to take a picture or to start/stop a sequence is provided to the camera module.

Furthermore, the base station runs a Graphical User Interface (GUI) to show all the information to the user/pilot. Following the SA paradigm, qualitative explanations are provided

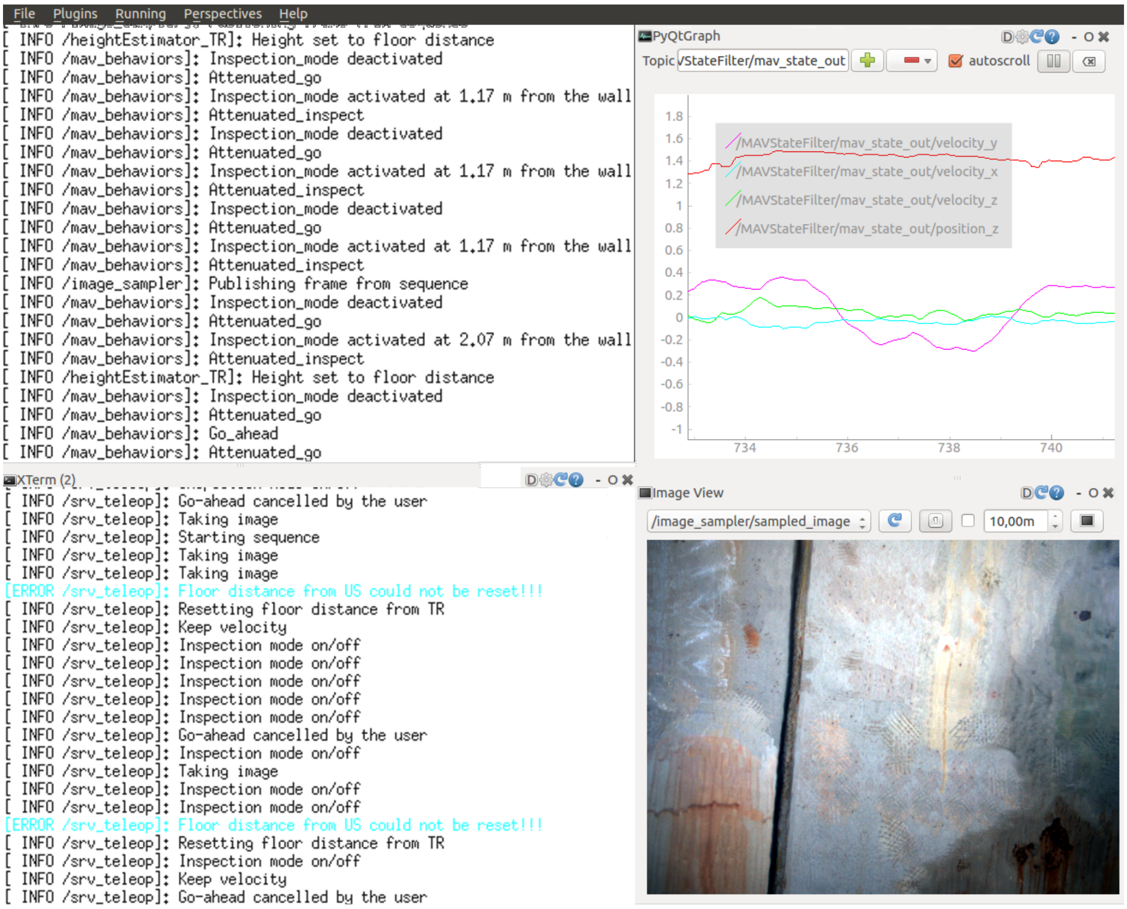


Figure 3.20: Graphical user interface.

to indicate what is happening during the course of a mission. For example, the GUI indicates “going forward” when the *go_ahead* behaviour is enabled, or “low battery landing” when the corresponding behaviour is activated. The GUI is also used to provide instructive feedback including the distances to the obstacles situated around the platform, the flight height and the estimated velocities. When using the optical flow sensors (pipelines for the SS1 and the SS3), the GUI also indicates the mode used to combine the information provided by the two optical flow sensors (see Tables 3.3 and 3.4). Finally, the user interface is also used to show the images captured with the on-board camera when these are requested by the user. Different visualization tools included in ROS, such as *rqt_image_view* or *rqt_plot*, have been used to develop the GUI. A screenshot of the final result is provided in Fig. 3.20.

3.7 Experimental Evaluation

This section provides the experimental assessment of the MAV. Since different configurations have been developed (using different sensor suites), this section firstly checks the flying ca-

pabilities of the different setups. In this regard, Section 3.7.1 presents several experiments performed to evaluate the state estimation and control in hovering and displacement manoeuvres. Secondly, in Section 3.7.2, the performance of the different robot behaviours is evaluated, showing how each one of them contributes to the control and/or safety of the platform. In third place, in Section 3.7.3, the performance/usability of the platform is evaluated during an inspection mission. Finally, the performance of the localization methods used for tagging the images is evaluated in Section 3.7.4. During the experiments, a motion capture system has been used to obtain the position, orientation and velocity of the platform. This is used as the ground truth (GT) to evaluate the performance of our system.

3.7.1 Hovering and Displacement Capabilities

In a first kind of experiments, the vehicle hovering capability has been assessed. This manoeuvre becomes a key component within the SA approach, as this is the reaction of the aerial platform while flying and waiting for new commands. In this regard, Fig. 3.21 provides some results obtained when using the SS1 fitted on-board the AscTec Firefly. This figure plots the histograms of the speed values during a 1-minute hovering manoeuvre performed in each of the four estate estimation modes (see Table 3.3). Indeed, each plot compares the histogram of the speeds measured using the motion capture system (using a continuous line) with the values estimated using the optical flow sensors (using dashed lines). To facilitate the comparison, the histograms have been generated using the same quantization bins, and they are provided as a probability. As can be observed, all the histograms are approximately zero-centered, what indicates that the platform performs a suitable hovering using the different state estimation modes. This histograms also show the quality of the on-board velocity estimations. Figure 3.22 shows the 3D position of the platform provided by the motion capture system during the four hovering flights. Notice that the deviations with regard to the first position which can be observed are normal since we do not apply position control but velocity control.

The hovering manoeuvre has been repeated using the laser scanner based system (i.e. the SS2) on-board the AscTec Pelican platform. The results are provided in Fig. 3.23. As already happened with the other platforms, the histograms resulting from the measured velocities are approximately zero-centered, and the displacement of the platform is pretty reduced.

In a second kind of experiments, the behaviour of the MAV has been evaluated while the user provides displacement commands. To be precise, the user/pilot is consigned to try to perform a square-like trajectory. We have proceeded in the same way as for the hovering experiments, so that four flights have been performed to evaluate the SS1 performance, using a different state estimation mode in each flight. For the first flight, the state estimation mode 0 has been used, so that the bottom-looking sensor has been utilized to estimate all the MAV velocities regarding the ground. The square-like trajectory has thus been performed in the XY plane. The rest of the flights, using the state estimation modes 1, 2 and 3, have been performed in front of a vertical wall, since this is required for the speed estimation. In these

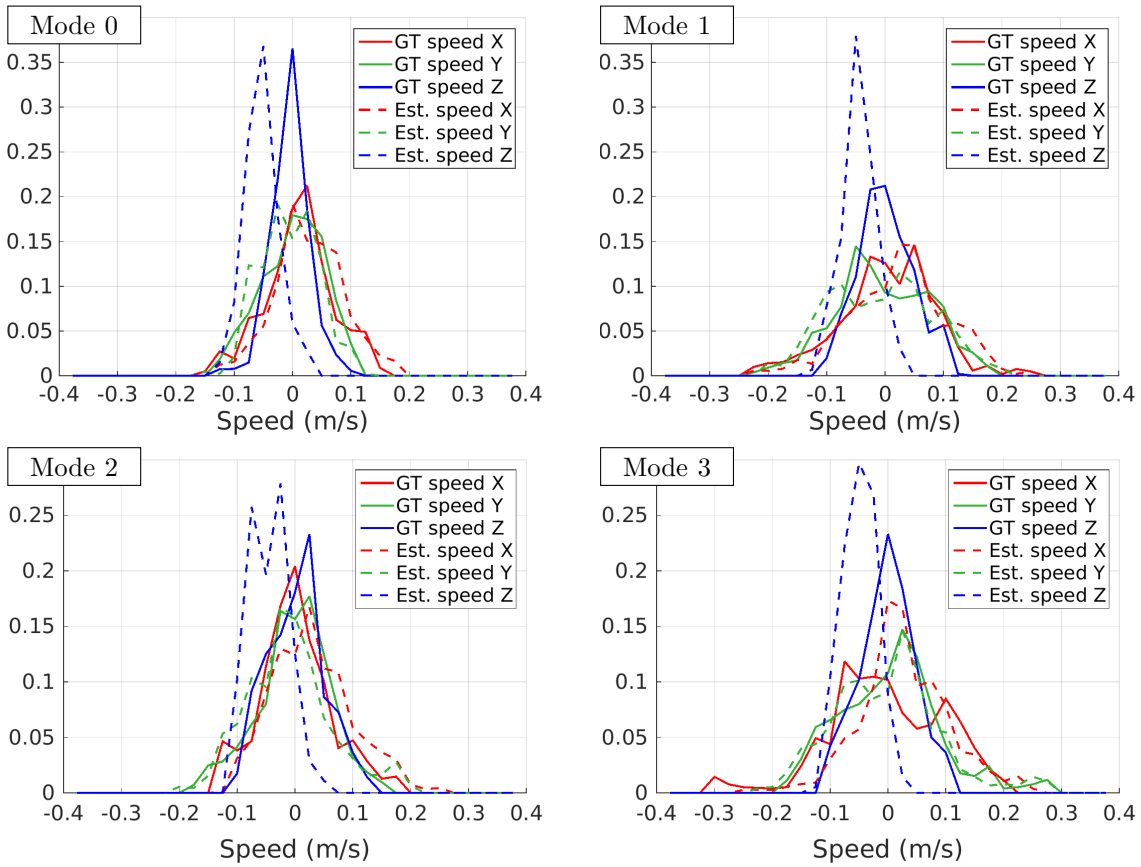


Figure 3.21: Normalized histograms of estimated speeds for 1-minute hovering flights performed using the SS1 and the different state estimation modes. Continuous lines are used for the data provided by the motion capture system (ground truth), while dashed lines are used for the values estimated by the MAV. Experiments performed using the AscTec Firefly.

flights, the square has been performed in the YZ plane, i.e. parallel to the wall. Figure 3.24 provides the trajectories performed by the MAV as indicated by the motion tracking system. As can be observed, the user/pilot can easily perform the square-like trajectory parallel to the reference frame. Remember that the system lacks of position control loop, so that the human is in charge of the positioning of the MAV.

These experiments also allow checking the vehicle reaction to the user commands. In this regard, Fig. 3.25 shows the user commands sent to perform the square-like trajectories (blue), together with the estimated velocities (red). Notice that, during these experiments, the vehicle has been operated far from obstacles, so that there are not attenuations nor repulsions, and the speed command provided by the MAV behaviours module coincides with the user desired speed. As can be observed in the plots, the estimated speeds follow the user desired speed, what indicates a successful operation of the velocity controllers. The plots also allow validating the suitability of the velocity estimation procedure, since the estimated velocities are compared with the velocities provided by the motion tracking system (green).

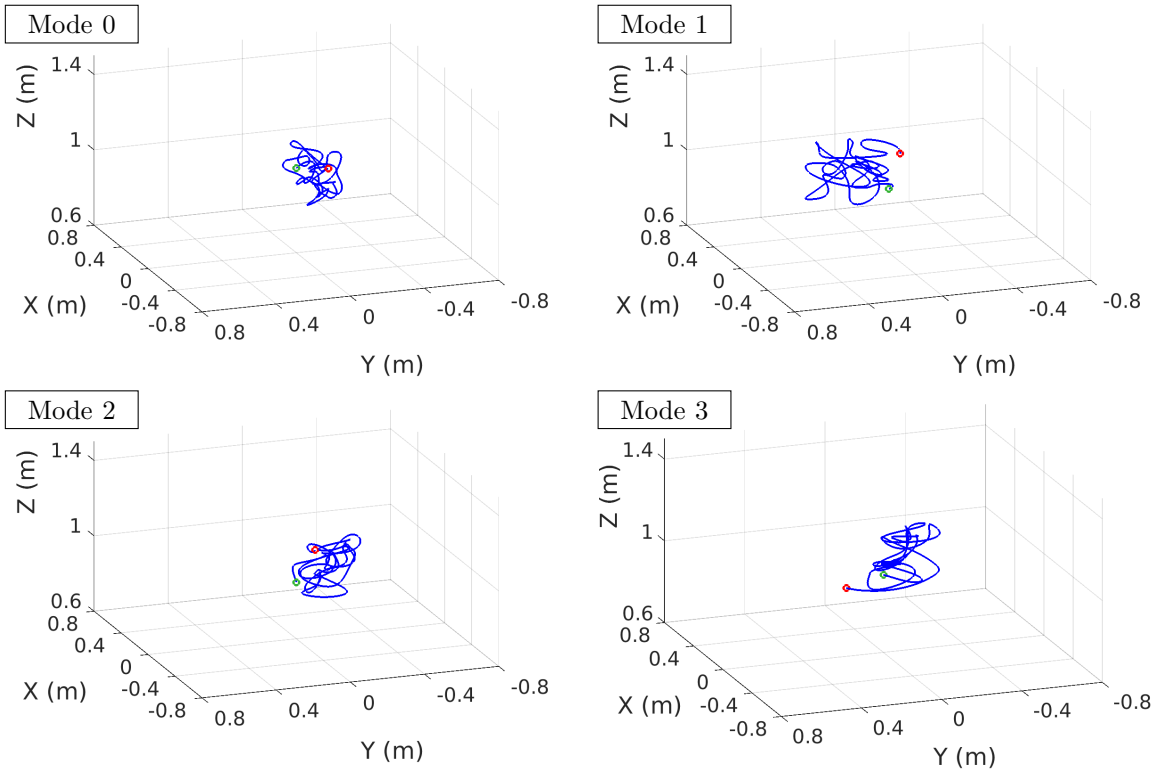


Figure 3.22: Plots of the position of the MAV indicated by the motion tracking system during 1-minute hovering flights performed using the SS1 and the different state estimation modes. Experiments performed using the AscTec Firefly. The green dot indicates the initial point, while the red dot indicates the final point.

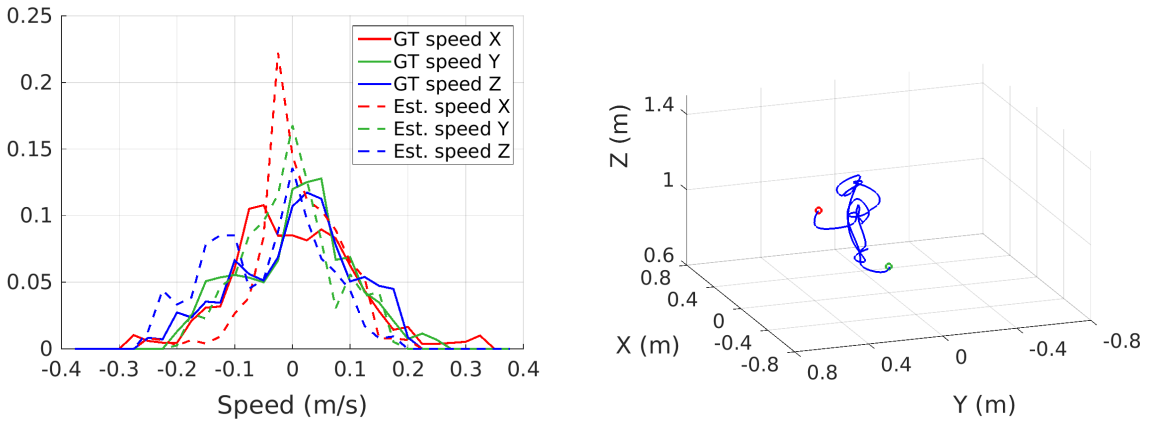


Figure 3.23: Results for a hovering flight using the SS2 on board the AscTec Pelican: (left) normalized histograms of estimated speeds (continuous lines are used for the data provided by the motion capture system, while dashed lines are used for the values estimated by the MAV), (right) position of the platform (the green and red dots indicate the initial and final points respectively).

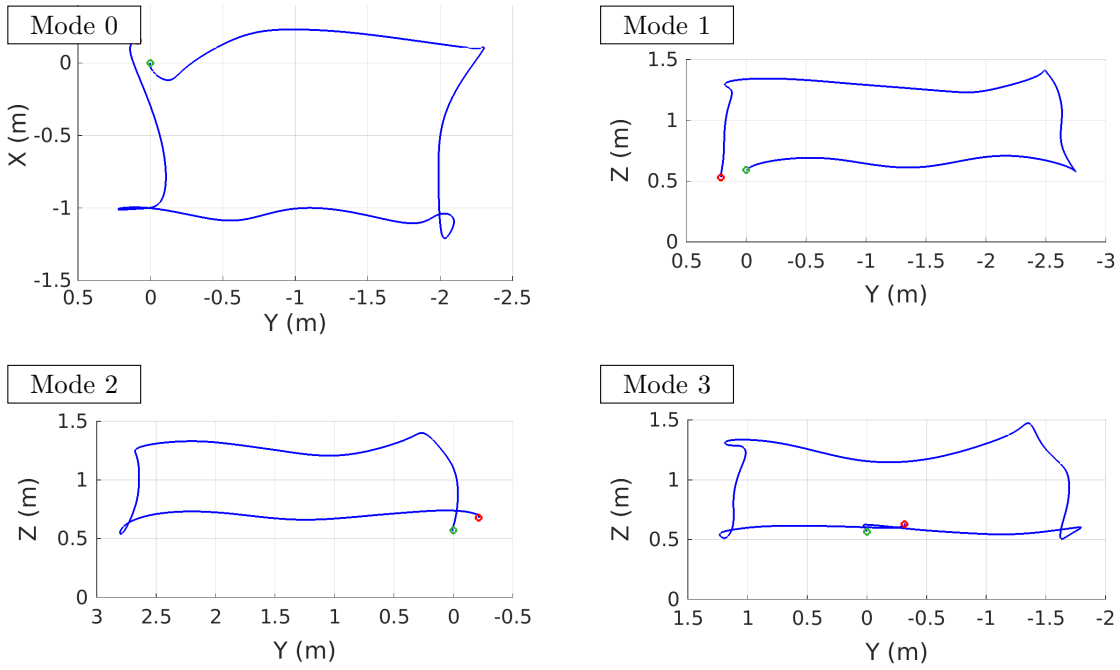


Figure 3.24: Plots of the trajectory of the MAV indicated by the motion tracking system during square-like flights performed using the SS1 and the different state estimation modes. Experiments performed using the AscTec Firefly. The green dot indicates the initial point, while the red dot indicates the final point.

A similar experiment has been performed using the platform equipped with the SS2. In this case, the trajectory followed by the vehicle consists in two consecutive squares performed at different heights. The plots corresponding to this experiment are provided in Fig.3.26.

Regarding the other platform fitted with the SS1, i.e. the AscTec Hummingbird, as already mentioned, it has been used as initial test bench to validate all the control and state estimation systems/modules. Its performance has resulted similar to the one exhibited by the AscTec Firefly. Nevertheless, this can not be evaluated using the motion tracking system since the IR light emitted by this device severely interferes with the Teraranger, used as optical height sensor. Notice that the Lidar-Lite optical range sensor installed on the other MAVs makes use of an IR laser to estimate the distance, so that it is not affected by the light emitted by the motion capture system. To check the operation of the Teraranger sensor, a hovering has been performed with the Hummingbird without using the motion capture system. The values measured during this flight are reporter in Fig. 3.27. The left plot provides the height measures while the right plot shows the vertical speed estimations. As can be observed, the results are consistent since the reduced variations in the flight height entail a vertical speed in the corresponding direction.

To assess the performance of the SS3 state estimation, a specific experiment has been

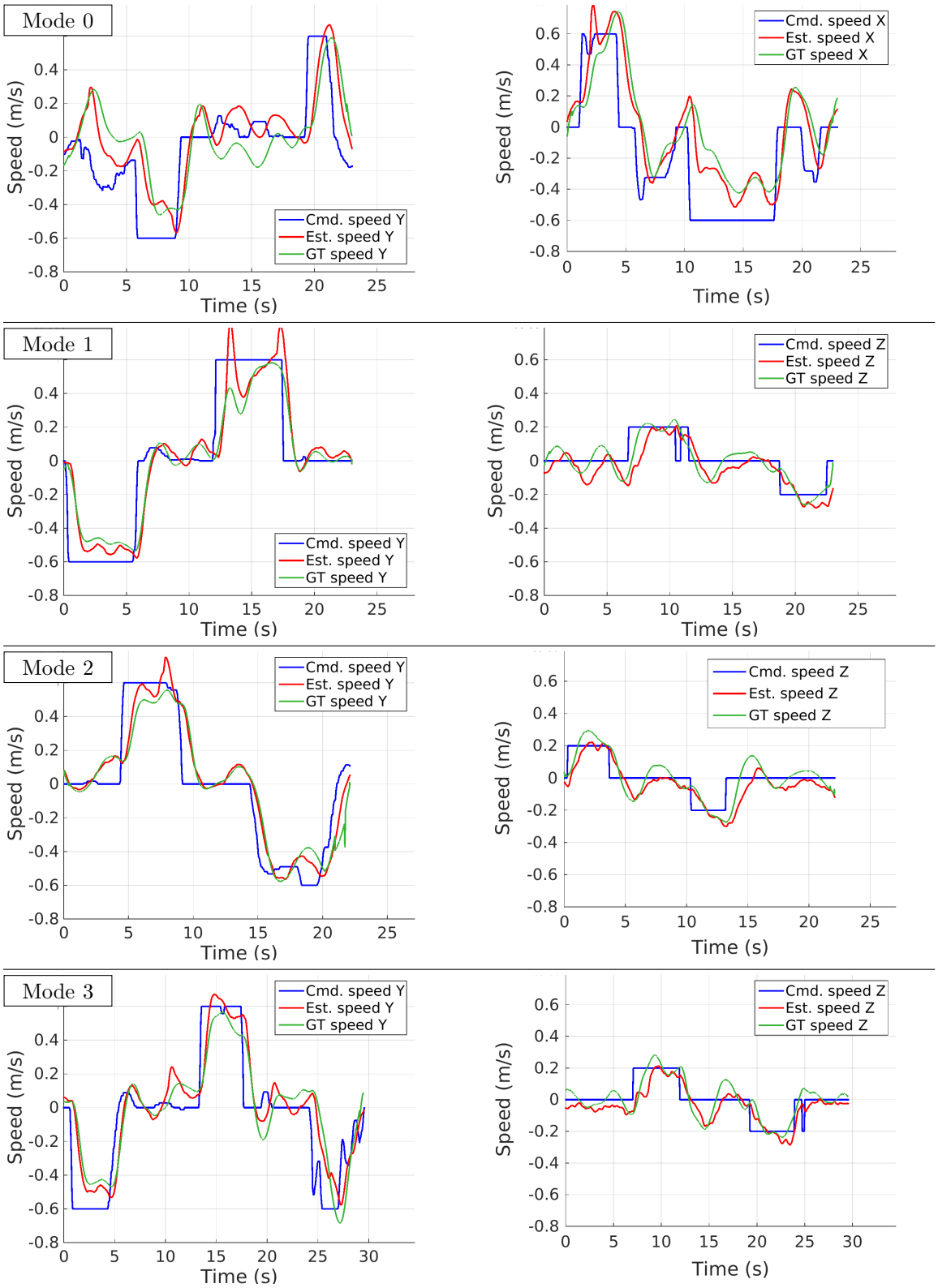


Figure 3.25: Plots of the speed of the MAV while receiving commands to perform square-like trajectories. Experiments performed using the SS1 on the AscTec Firefly.

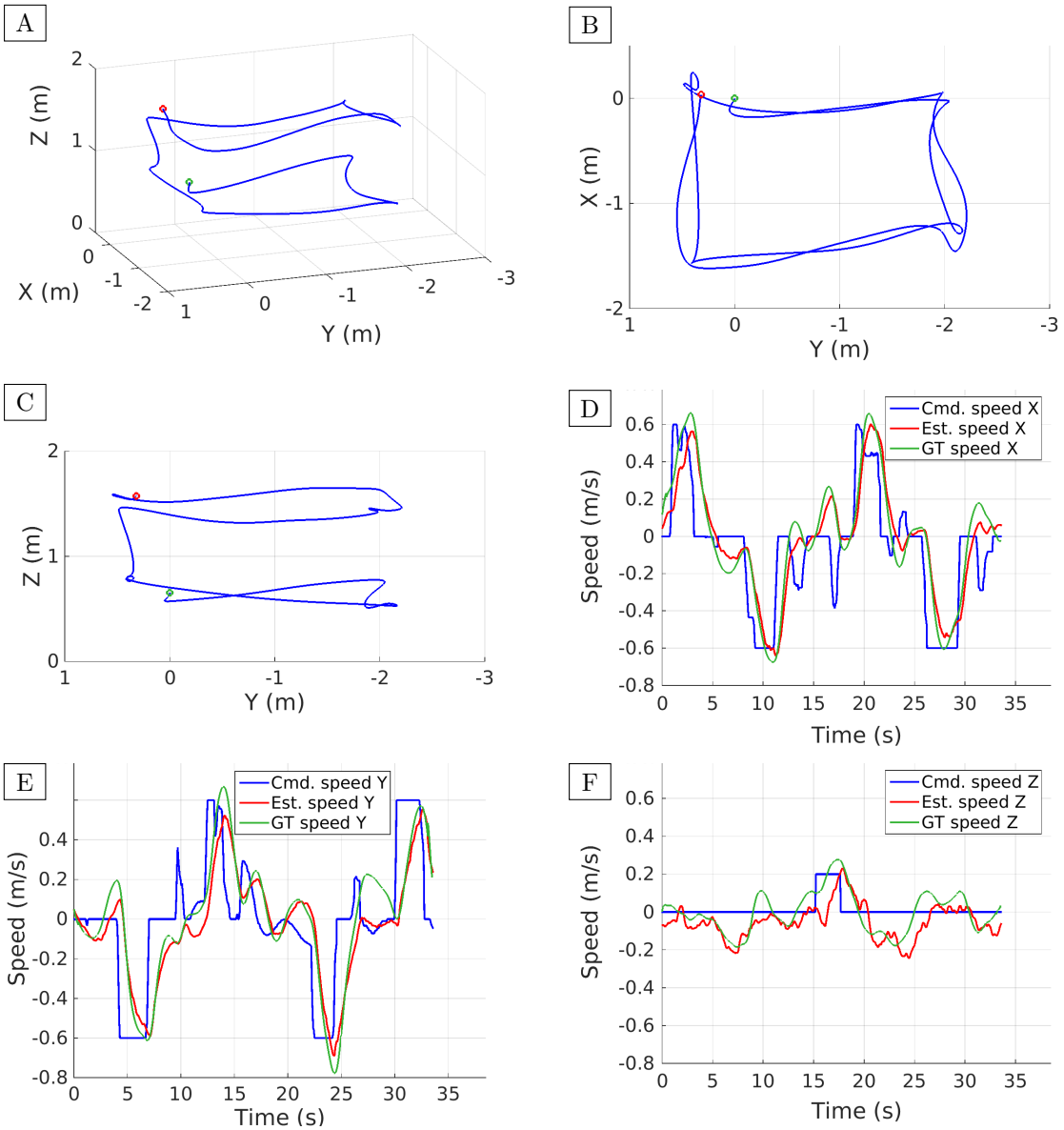


Figure 3.26: Results obtained with the AscTec Pelican fitted with the SS2, commanded to perform a double-square trajectory: (A) plot of the trajectory indicated by the motion tracking system (the green and red dots indicate the initial and final points), (B-C) 2D projections of the trajectory, (D-F) reactions of the MAV to the velocity commands in the tree axes.

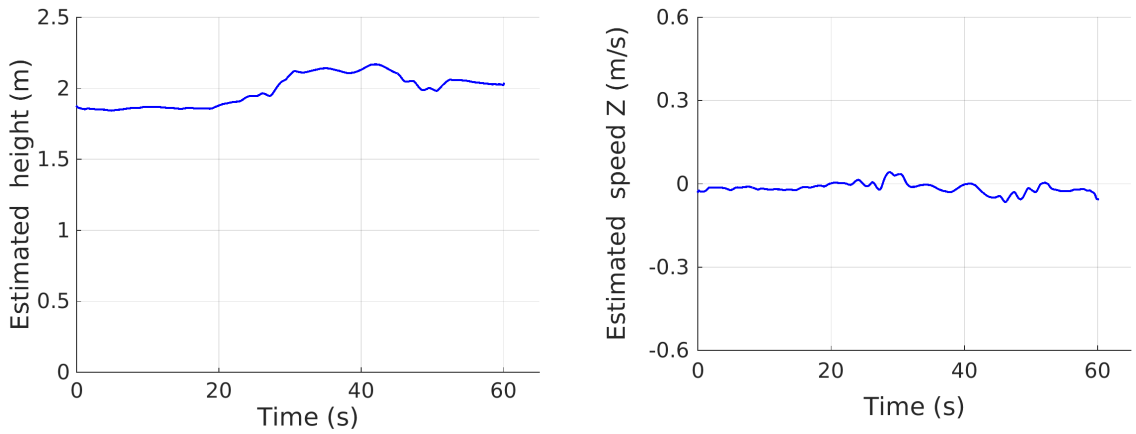


Figure 3.27: Height (left) and vertical speed (right) measured using the Teraranger sensor during a 1-minute hovering flight.

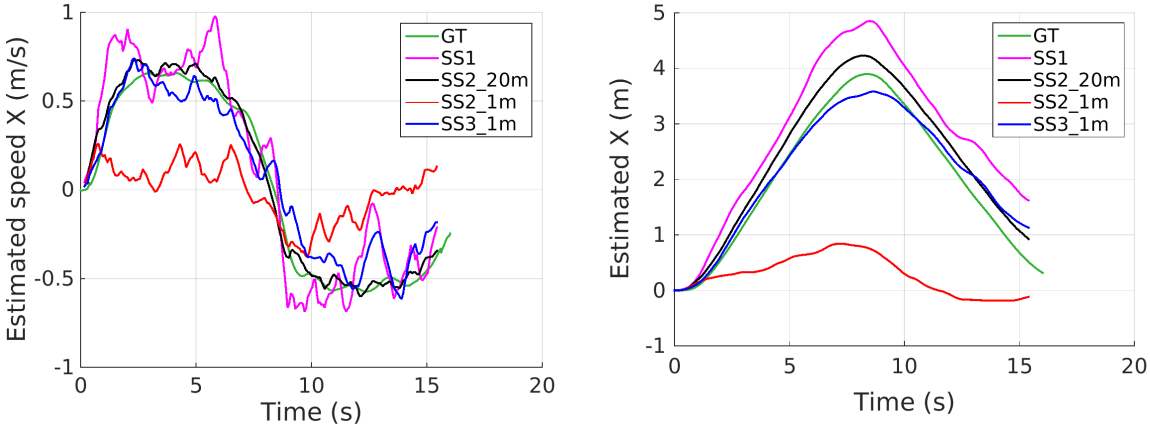


Figure 3.28: Results for a flight parallel to a wall using the SS3: (left) estimated speeds, (right) positions estimated via speed integration. The results are compared with the ground truth and the values obtained using the SS1 and SS2. Experiment performed limiting the laser scanner range to 1 m to force the situation inside the laboratory.

carried out. This consists in flying the AscTec Pelican platform forwards parallel to a wall situated at its left. The vehicle has been displaced around 4 meters, and then it has been moved backwards approximately to the initial location. During this flight, all the data provided by the sensors comprising the SS3 have been saved. Then, several executions using the different state estimation pipelines have been carried out. Firstly, the SS1 and SS2 pipelines have been used to estimate the vehicle speed. Figure 3.28 [left] provides the obtained results in pink and black respectively. As can be observed, these approximately follow the ground truth value provided by the motion tracking system, indicated in green.

Then, the SS2 pipeline has been used once again, now limiting the laser scanner maximum range to 1 m (the sensor detects obstacles at 20 m) in order to restrict the readings to the left wall, when estimating the vehicle velocities. In other words, the rest of the walls and structures

in the laboratory are ignored by the MAV. Under these conditions, the SS2, which relies solely on the laser scanner to estimate its longitudinal velocity, is not able to provide a correct estimation, as shown in Fig. 3.28 [left] in red. A last execution for the same sensor data has been performed using the SS3 pipeline. As can be observed in blue, the speed estimated by the laser odometer, when the optical flow data is used as initial guess, successfully approximates the ground truth speed.

Figure 3.28 [right] provides an additional analysis of the results obtained with this experiment. In this figure, the estimated speeds have been integrated to obtain an estimate of the vehicle position along the X axis. As can be observed, when the laser scanner range is limited, the vehicle position indicated by the SS3 (blue) approximately matches the position indicated by the motion tracking system (green), while the displacement indicated by the SS2 (red) is clearly underestimated, as could be expected.

3.7.2 Robot Behaviour Evaluation

Once we have assessed the flight capabilities of the MAVs equipped with the different sensor suites, we proceed to evaluate the performance of the robot behaviours. In the following, several experimental results are reported in this regard, where each behaviour is evaluated using only one of the MAVs (and a specific sensor suite). Similar results have nevertheless been observed for the other platforms/equipment.

In a first experiment, we check how the platform behaves in a situation of imminent collision. To do that, we move the Firefly platform equipped with the SS1 towards a wall. The plot for this experiment is provided in Fig. 3.29. The right plot shows how the longitudinal speed command provided by the *MAV_behaviours* module ($\dot{x}_d$) coincides with a user command ($\dot{x}_{ud}$) of around 0.4 m/s until the wall in front of the vehicle becomes closer than 1.5 m (instant A), moment at which the user-desired velocity is attenuated by the *attenuated_go* behaviour making the speed command decrease in accordance to the closeness to the wall. When the wall becomes closer than 1 m (instant B), which is the minimum distance allowed (d_m), the user-desired speed is completely cancelled by the *prevent_collision* behaviour, and the platform stops. Notice that the user desired speed is still around 0.4 m/s until instant C. The left plot provides the vehicle trajectory and the wall position, as captured by the motion tracking system. The actuation of the *attenuated_go* behaviour is indicated in a different colour.

A second experiment, reported in Fig. 3.30 [right], checks the performance of the *go_ahead* behaviour. In this experiment, we have used the Pelican platform fitted with the SS2. At the beginning, the user indicates a longitudinal desired speed of 0.4 m/s and then activates the *go_ahead* behaviour (instant A). At this moment, in accordance to the behaviour definition, the speed command produced by the *MAV_behaviours* module ($\dot{x}_d$) keeps at 0.4 m/s although the user-desired speed ($\dot{x}_{ud}$) returns to zero. This value is kept until the wall in front of the vehicle becomes closer than d_m (instant B), which is set to 1.2 m for this experiment. Then, the *prevent_collision* behaviour cancels the *go_ahead* command and stops the platform. This

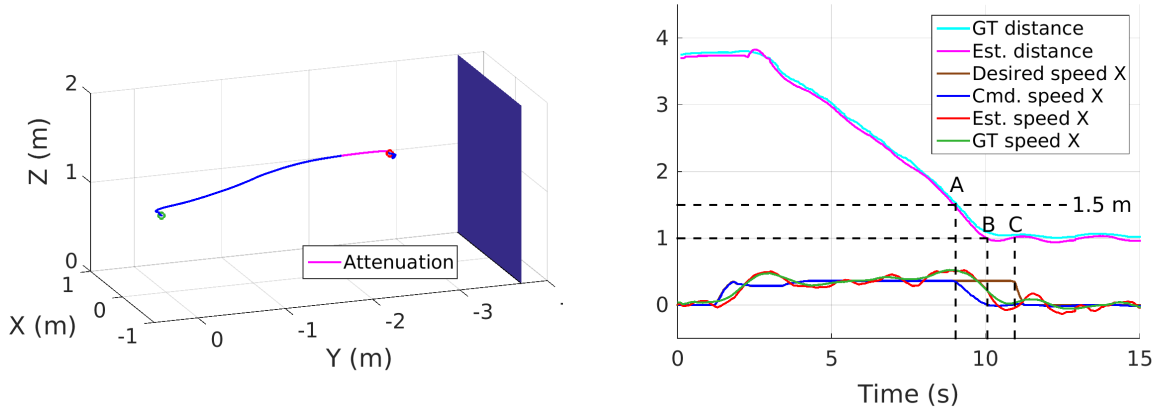


Figure 3.29: Performance of the *attenuated_go* and the *prevent_collision* behaviours: (left) vehicle trajectory and wall position indicated by the motion capture system, (right) the user-desired speed is obeyed (→A), it is attenuated (A→B) and cancelled to prevent an imminent collision (B→C) until the user-desired speed does become zero (C→). All units are in SI (m or m/s accordingly).

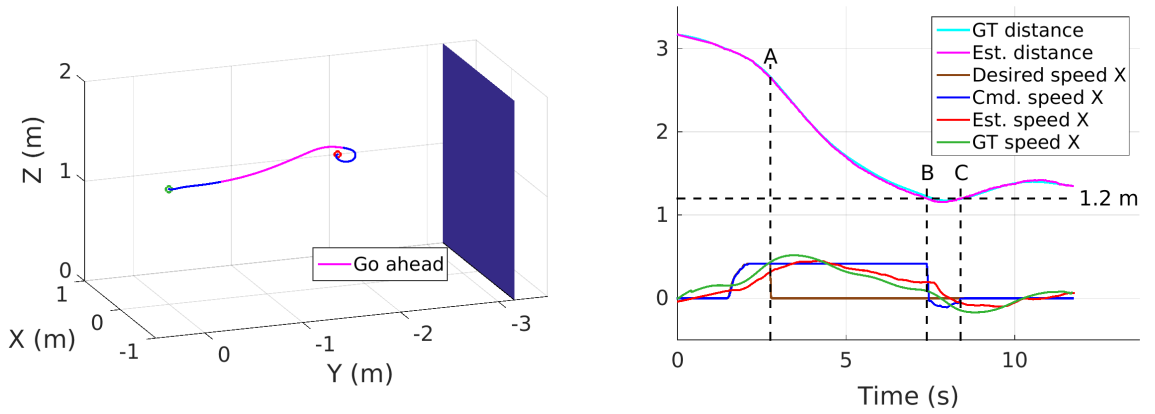


Figure 3.30: Performance of the *go_ahead* and the *prevent_collision* behaviours: (left) vehicle trajectory and wall position indicated by the motion capture system, (right) the user-desired speed is sustained while the wall is at enough distance (A→B), it is cancelled and even forced to be negative to prevent an imminent collision (B→C) until the platform is again at the safe distance (C→). All units are in SI (m or m/s accordingly).

behaviour is also in charge of producing the negative speed command that separates the platform from the wall until it is again at the safe distance (instant C). Figure 3.30 [left] shows the vehicle trajectory, indicating when the *go_ahead* behaviour is active.

In a third experiment, we check the performance of the *limit_max_height* behaviour. Figure 3.31 [right] shows how the Firefly platform equipped with the SS1 ascends following the vertical user-desired speed $\dot{z}_{ud}$ (and the vertical speed command $\dot{z}_d$) until the platform reaches a height of 3 m (instant A), which was set as the maximum height for this experiment (z_M). From time instants A to B, the behaviour prevents the platform from going higher

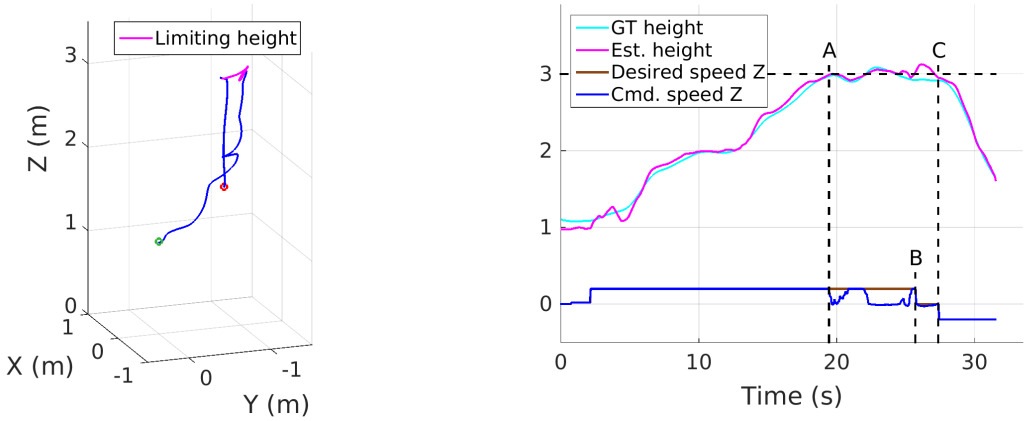


Figure 3.31: Performance of the *limit_max_height* behaviour: (left) vehicle trajectory indicating when the flight height is limited, (right) the user-desired vertical speed is obeyed until the platform reaches the maximum allowed height (→A), then the desired vertical speed is ignored (A→B) until it becomes zero (B→C), and finally the platform descends following again the desired speed (C→). All units are in SI (m or m/s accordingly).

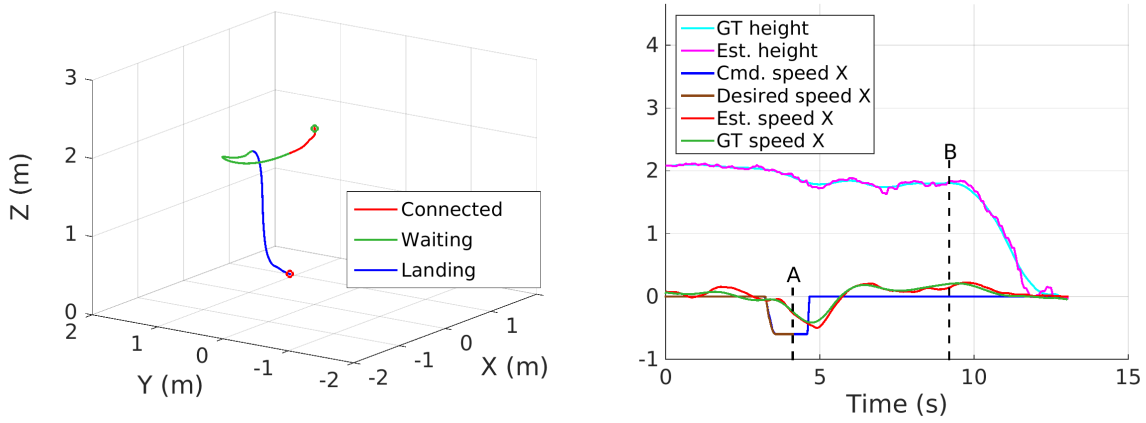


Figure 3.32: Performance of the *waiting_for_connectivity* behaviour: (left) vehicle trajectory indicated by the motion capture system, (right) the communication with the base station is lost during the flight (→A), what makes the behaviour force a hovering manoeuvre (A→B) while the vehicle tries to reconnect; after waiting for five seconds without success, the behaviour makes the platform land (B→). All units are in SI (m or m/s accordingly).

ignoring the vertical user-desired speed until it becomes zero (instant B). Next, the platform descends since the user asks for a negative vertical speed (instant C). Figure 3.31 [left] shows the vehicle trajectory, indicating when the flight height is being limited.

Results for a fourth experiment are plotted in Fig 3.32 [right]. This case involves the *waiting_for_connectivity* behaviour and the Firefly platform. At the beginning of the experiment, the user orders a negative longitudinal speed to move the platform. During the displacement, the communication with the base station is lost (instant A), so that the user-desired speed signal $\dot{x}_{ud}$ is no longer available at the platform. As a consequence, the behaviour takes

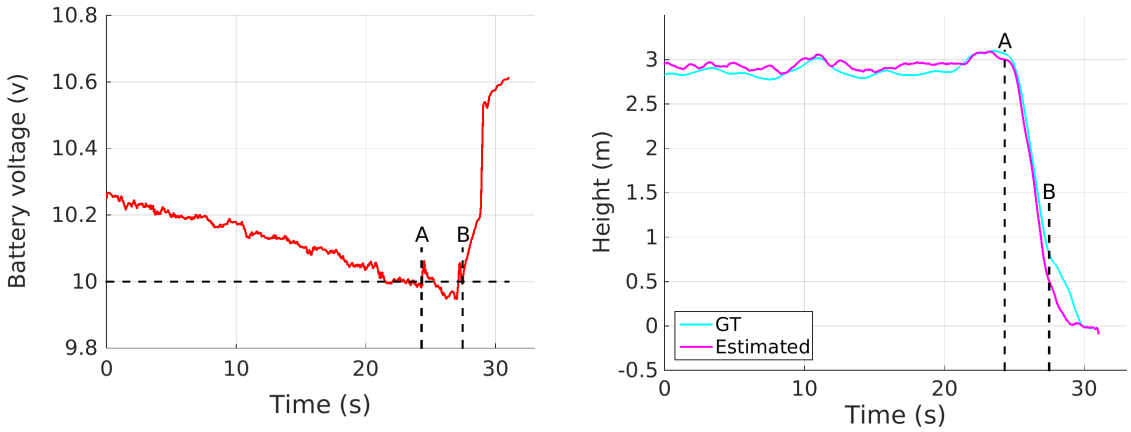


Figure 3.33: Performance of the *low_battery_land* behaviour: the platform hovers at 3 m until the battery voltage is below 10 V (→A), what makes the behaviour initiate the descending (A→B) and landing manoeuvres (B→).

control and makes the vehicle hover while waits for a reconnection. After 5 seconds (instant B), the communication link has not been restored and the behaviour decides to make the platform land. Figure 3.32 [left] shows the vehicle trajectory, where different colours are used to indicate when the vehicle is receiving the desired command, when it is waiting and trying to reconnect, and when the vehicle performs the landing manoeuvre.

Figure 3.33 corresponds to a fifth experiment, aiming at checking the performance of the *low_battery_land* behaviour. During this experiment, the Pelican is left hovering at almost 3 m until the battery voltage becomes lower than v_m, which is set to 10 V (instant A). At this moment, the behaviour takes control of the platform to make it land. The landing manoeuvre starts with a *descending* stage, to make the platform reduce its height up to 0.5 m, and finishes with the deceleration of the motors thrust (instant B).

Figure 3.34 describes a sixth experiment aiming at checking the performance of the *ensure_reference_surface_detection* behaviour. This behaviour is only used with the SS1, so this experiment has been performed with the Firefly platform. The experiment starts with the platform flying at a certain distance from the front wall, so that the vehicle only makes use of the ground-looking optical flow sensor to estimate its velocity (state estimation mode 0). Within this mode, the vehicle is allowed to ascend (action 1) until the maximum distance to the ground, the reference surface, is attained (the maximum distance d_{b_M} was set to 1.5 m for this experiment). The next ascending orders (action 2) are ignored. Next, the vehicle moves towards the front wall (action 3) until the vehicle is close enough so as to also use this wall to estimate its state (state estimation mode 1). Once the vehicle is close to the wall, it moves upwards (action 4) until the ground becomes too far for the ground-looking optical flow sensor (2 m for this experiment) and the front wall becomes the only reference surface (state estimation mode 2). Subsequently, the user tries to turn the vehicle to the right (action 5) but this action is ignored to ensure the reference surface detection. Figure 3.35 [A] shows the

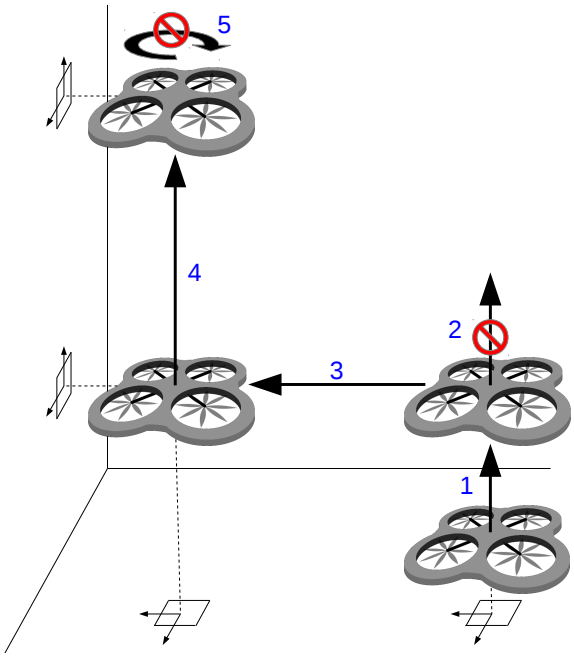


Figure 3.34: Experiment to illustrate the performance of the *ensure_reference_surface_detection* behaviour. Only movements that guarantee the detection of at least one reference surface are allowed.

trajectory followed by the MAV, indicating the estimation mode used. Figures 3.35 [B-D] plot sensor data for the full operation, and for, respectively, the longitudinal, vertical and angular speeds. The distance to the front wall and the vehicle height are shown to make evident the corresponding motion.

3.7.3 Illustration of an Inspection Mission

In a last experiment involving specific behaviours, we show the performance of the platform during an inspection task. This experiment has been performed using the Pelican platform fitted with the SS2, and a 2.5×4 m canvas, which has been printed to simulate the metallic plates of a vessel wall (see Fig. 3.37 [A]). The operation starts when the user/surveyor makes the platform approach the wall under inspection, where the canvas has been situated. At more or less 1 m distance, the *inspection mode* is activated, and, hence, longitudinal motion as well as rotations in yaw are not allowed to ensure better image capture conditions. The operator next orders lateral and vertical motion commands to sweep the surface, while records an image sequence at 10 Hz. Figure 3.36 [A] shows the vehicle trajectory, indicating when the inspection mode is active. Figures 3.36 [B-C] illustrate the full operation for the longitudinal [B], lateral [C] and vertical [D] motions. These velocities are shown at the bottom of the plots, while distances to, respectively, the front wall/left wall/ground are shown at the top, to make evident the corresponding motion. Notice that, when the *inspection mode* is enabled

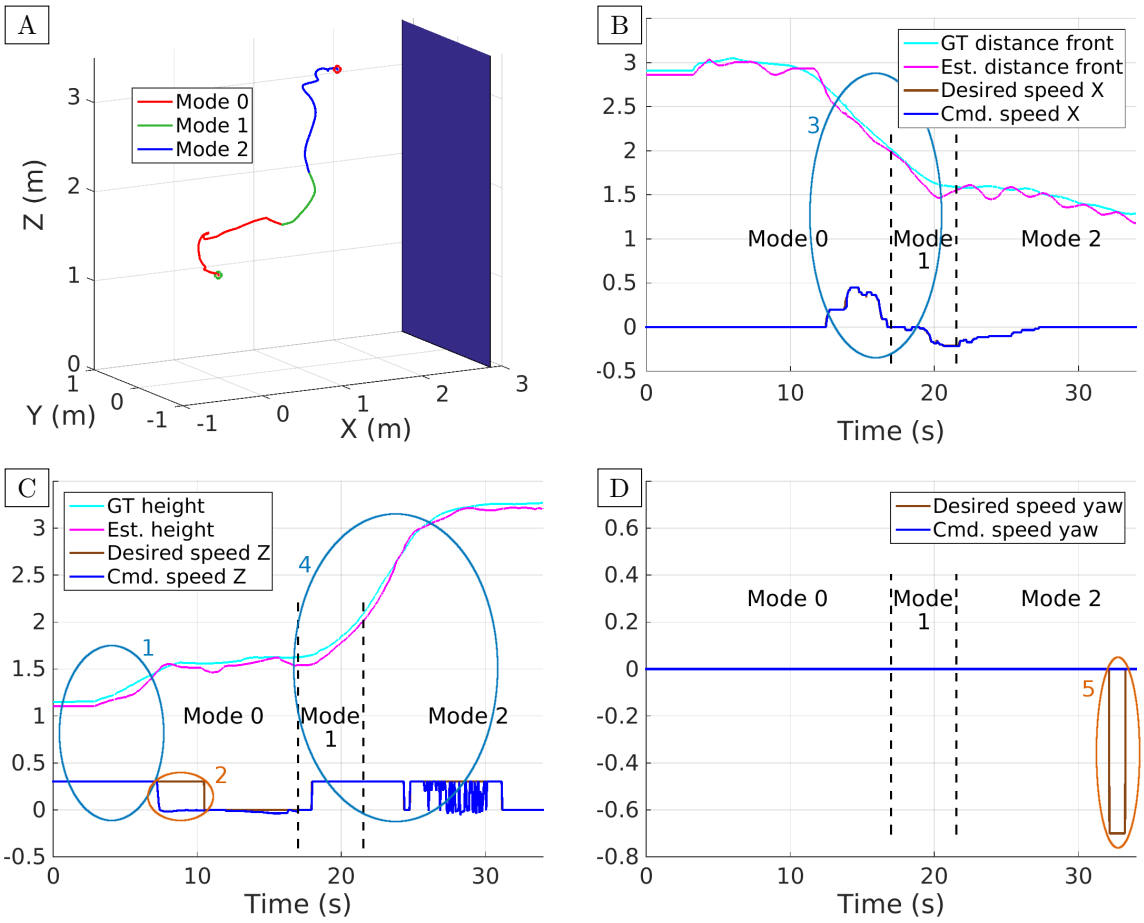


Figure 3.35: Illustration of the performance of the *ensure_reference_surface_detection* behaviour: (A) trajectory followed by the MAV indicating the estimation mode used, (B-D) longitudinal, vertical and angular commands/displacements (see text for the explanation). All units are in SI (m, m/s or rad/s accordingly).

(between instants A and B):

- the longitudinal user-desired speed is ignored, and a PID controller is in charge of keeping the distance to the inspected wall,

- the user only has the option of selecting hovering or motion in the vertical or lateral direction, but the speed command is set to ±0.2 m/s.

The plots also show repulsive speed commands produced when the platform is below 1 m regarding the front or left wall (see instants C, D and E). Figures 3.37 [B-C] show some of the pictures taken during the inspection. The portion of canvas captured in each picture is indicated in Fig. 3.37 [A].

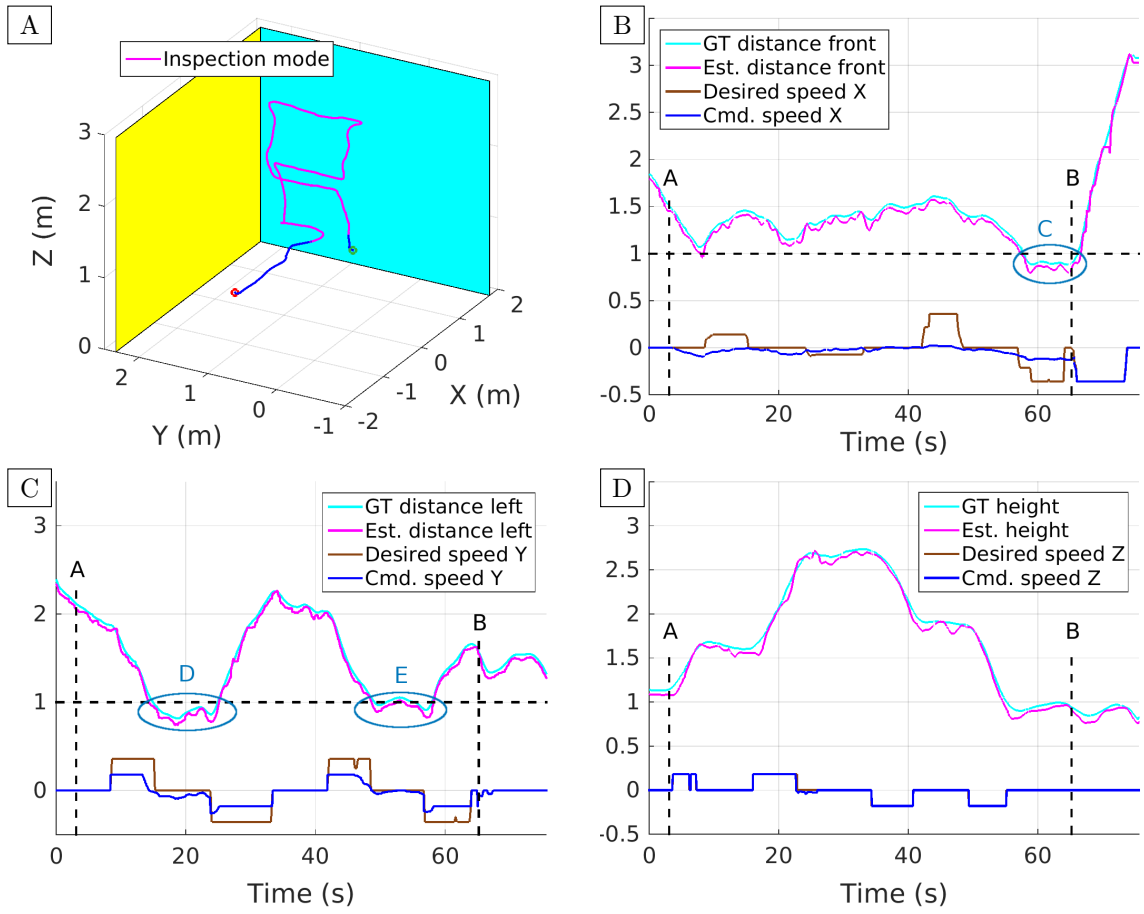


Figure 3.36: Performance of the platform during an inspection task using the *inspection mode*: (A) walls and vehicle trajectory, indicating when the *inspection mode* is active, (B-D) longitudinal, lateral and vertical commands/displacements (see text for the explanation). All units are in SI (m or m/s accordingly).

3.7.4 Position Estimation for Image Tagging

Finally, we have evaluated the SLAM algorithms as position estimators to tag the images taken with the MAV, during an inspection mission. To do that, the positions provided by the two SLAM methods have been compared with the position indicated by the motion capture system. Two experiments have been performed to this end. The first one, reported in Fig. 3.38, consists in a free flight including movements along the three axes. The second flight, reported in Fig. 3.39, consists in a sweeping of a wall, as in an inspection flight. As can be observed in the plots [A] and [B] included in these figures, the trajectories provided by both SLAM methods mostly coincide with the path provided by the motion tracking system.

To evaluate the position error of each SLAM method, we have computed the histogram of the difference regarding the ground truth value. This difference has been computed for the three axes separately. The resulting histograms are also provided in Figs. 3.38 and 3.39.

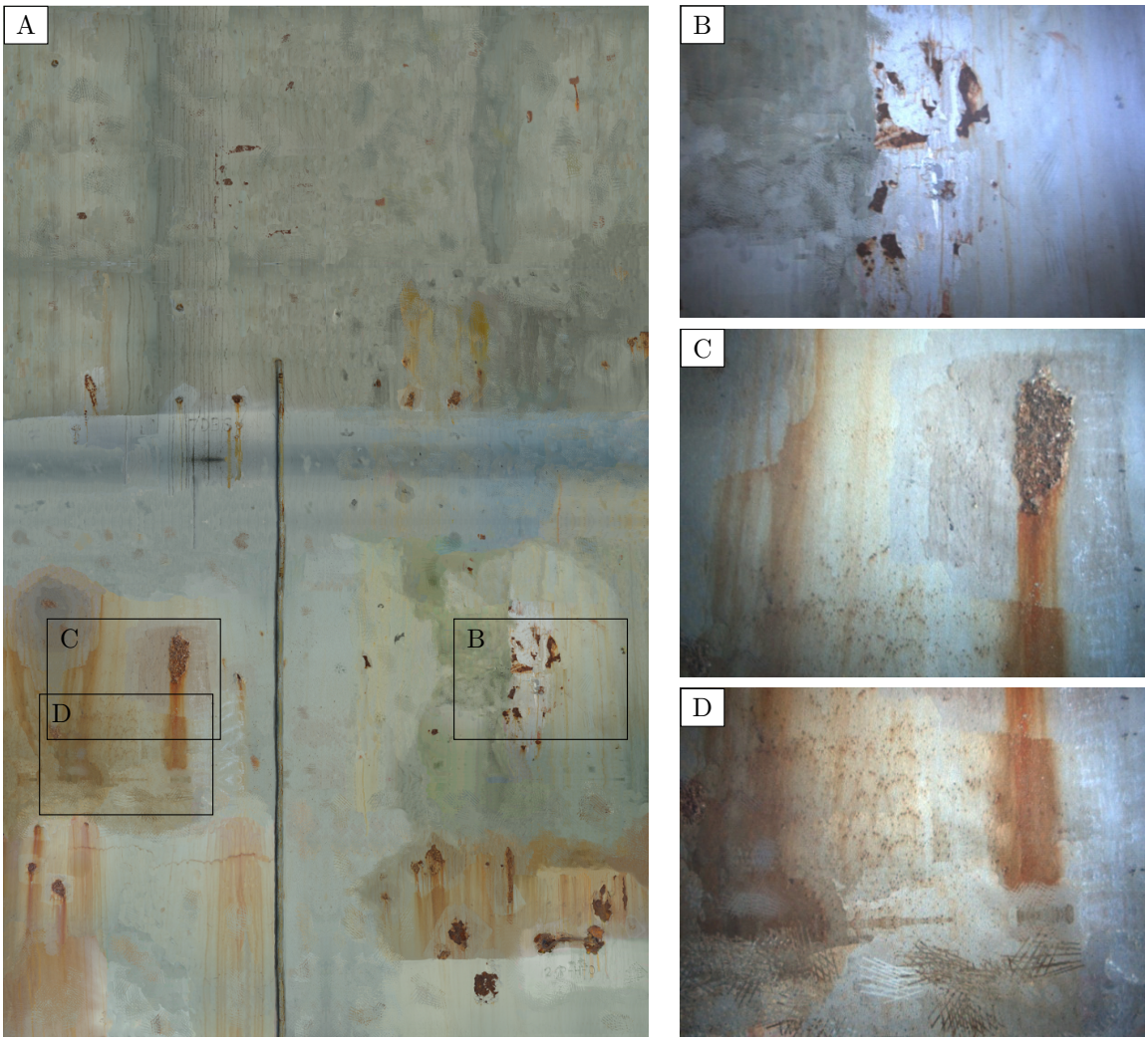


Figure 3.37: Pictures taken during the experiment to simulate an inspection mission: (A) canvas printed to simulate a vessel wall, (B-C) pictures taken using the MAV.

Subfigures [C] provide the error histograms corresponding to the ORB-SLAM method, while subfigures [D] provide the values obtained employing the GMapping method, used as part of the SS2 pipeline. As can be observed, the position error is in all cases small and zero-centred.

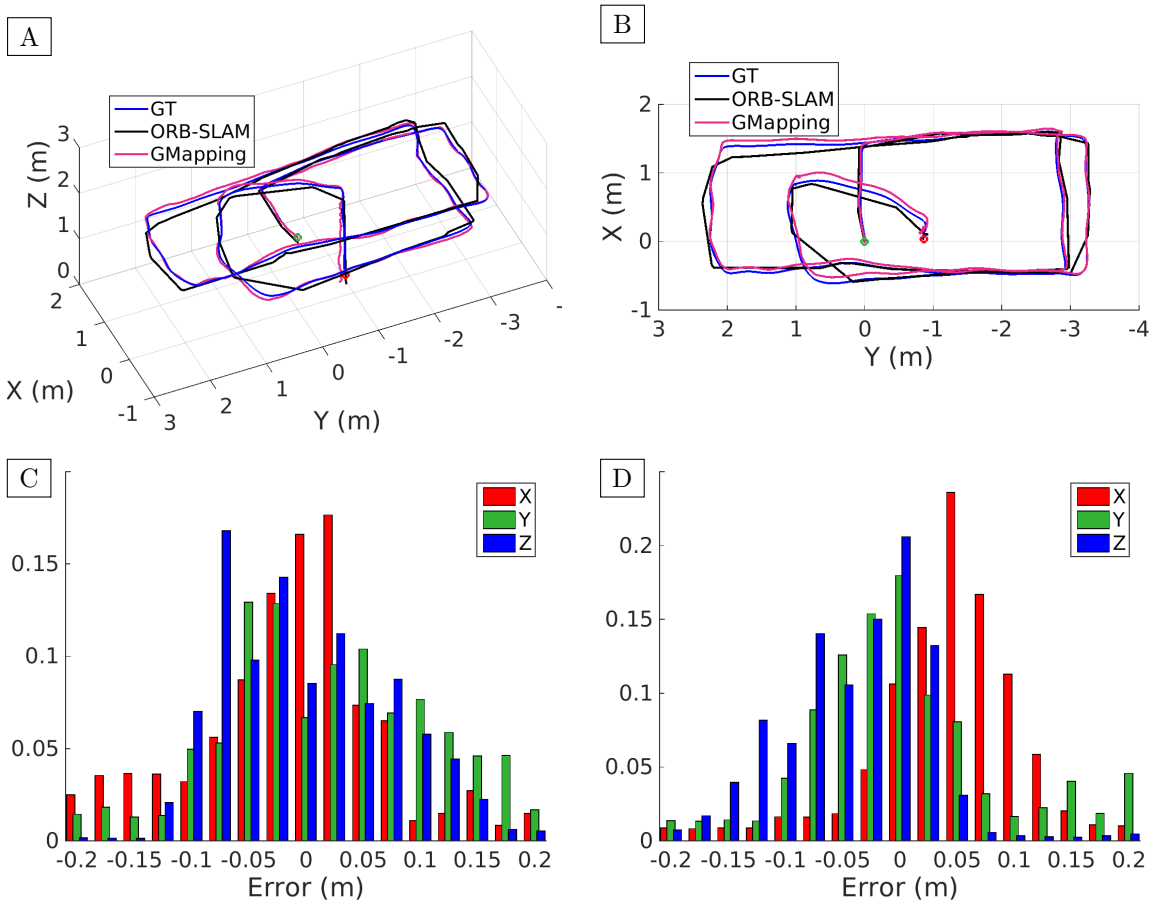


Figure 3.38: Performance of the SLAM algorithms in a first experiment: (A/B) trajectory estimated by the two SLAM algorithms, compared with the trajectory provided by the motion tracking system, (C) position errors for ORB-SLAM, and (D) position errors for GMapping.

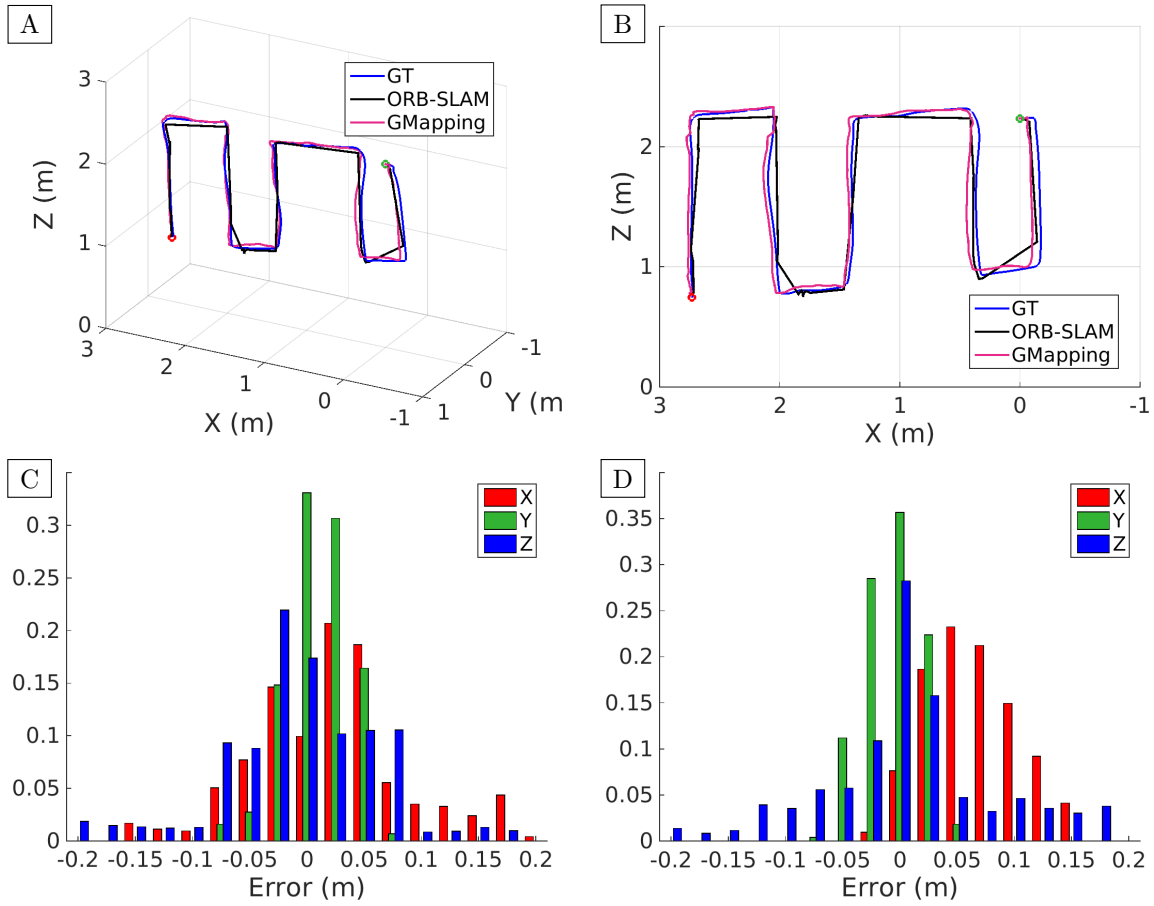


Figure 3.39: Performance of the SLAM algorithms in a second experiment: (A/B) trajectory estimated by the two SLAM algorithms, compared with the trajectory provided by the motion tracking system, (C) position errors for ORB-SLAM, and (D) position errors for GMapping.

Chapter 4

Vision-based Algorithms for Defect Detection on Vessels

This chapter provides novel methods for crack and corrosion detection in vessel structures. The methods presented are specifically devised for visual inspection, so that they are based solely on colour images. The chapter is organized in six sections. Section 4.1 discusses about different main approaches that can be useful for this particular problem. Section 4.2 details how the performance of the different proposed methods has been evaluated, describing the image datasets used for that purpose and detailing the metrics employed. Section 4.3 focuses on the detection of corrosion. It proposes a general structure for a corrosion detector, and explores different alternatives in order to improve the detection performance. This section provides details about the design, configuration and performance of the alternatives proposed. Section 4.4 presents an initial crack detection method which is later improved by combining it with the corrosion detector method presented in a previous section. This section also provides the set up details, the detection results and the experimental evaluation. An alternative defect detection approach is described in Section 4.5. It introduces the idea of using saliency for detecting generic defects on vessels structures. It proposes the use of two saliency-related features, and provides two different defect detection frameworks to combine the information that these features convey. The section also provides the evaluation of the performance of the resulting detectors, and states some conclusions. Finally, Section 4.6 presents the idea of combining a generic defect detector based on saliency, with specific corrosion and crack detectors, in order to improve the global detection performance.

4.1 General Discussion

Our detection problem consists in classifying every image pixel of an input image in one of two classes: *defect* or *non-defect*. The class defect can refer to the presence of cracks, corrosion or unspecific defects, depending on the case considered.

As seen in Sections 2.2.1 and 2.2.2, the existing image-based methods for crack or corrosion detection require specific conditions for proper performance. These conditions include, for example, a very short (and sometimes fixed) distance to the inspected surface, which may be difficult to accomplish during the inspection of the different vessel structures, specially if

a flying robotic platform is used for that purpose. For this reason, the corrosion and crack detectors developed within the context of this thesis have been devised to tolerate certain variability in the image capture conditions.

Furthermore, in this study, different *image processing* and *machine learning* techniques have been explored. On the one hand, by image processing techniques we mainly mean methods being applied to the input image, which is treated as a 2D signal. Within the context of a classification problem, the goal of these techniques is to emphasize certain properties of the image (or image patches), which allow their discrimination into the relevant classes.

On the other hand, the machine learning techniques we apply in this chapter belong to the category of *supervised learning*, and thus comprise two different stages. The first one, the so-called *learning stage*, consists in using a set of already classified images (or image patches) to train the algorithm so that this can learn the general rules which give rise to the correct classifications. In the second stage, the *classification stage*, the algorithm applies the learned rules to assign new samples to the class that best fits.

4.2 Experimental Setup

Three image datasets have been used during the development and performance evaluation of the defect detectors. These datasets contain pictures taken from vessel structures affected by some kind of defective situation, including corrosion and cracks. Each dataset will be used to evaluate a different set of detection algorithms:

- *Corrosion dataset.* It comprises 49 images including corroded surfaces, and is used in Section 4.3 to evaluate the corrosion detection algorithms.
- *Cracks dataset.* It comprises 23 pictures including cracks observed at different vessel structures, and which present different appearances. This dataset is used in Section 4.4 to evaluate the crack detection algorithms.
- *General defects dataset.* It comprises 73 pictures with different kinds of defects in vessel structures and surfaces, including different types of corrosion, cracks and coating breakdown. This dataset consists of all the images of the two other datasets and is used in Section 4.5, where the detection of unspecified defects on vessels is addressed.

The images in the datasets have been taken using several cameras, with different illumination conditions and with resolutions ranging from 492×379 to 2359×1582 pixels.

The datasets also include the ground truth images. These are binary images where the defective areas have been manually labelled in white. The ground truth images have been used to qualitatively assess the classification results produced by the defect detection algorithms. By way of example, Fig. 4.1 shows some of the images included in the different datasets, together with their respective ground truths.

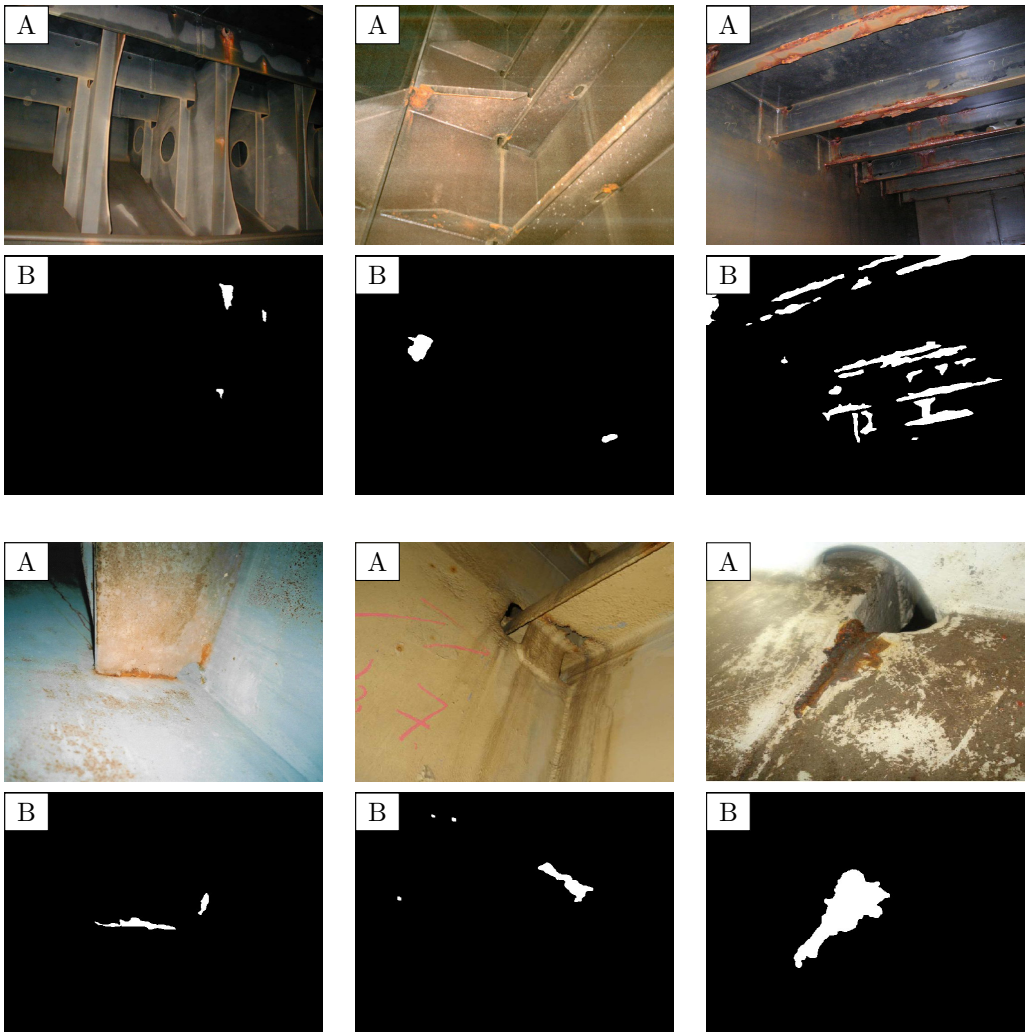


Figure 4.1: Some images from our datasets: (A) pictures taken of different vessel structures affected by corrosion and/or cracks, (B) hand-labelled ground truth images.

For a proper evaluation of machine learning techniques performance, we use different image datasets during the training and test stages. In this regard, we use the Leave-One-Out Cross-Validation (LOOCV) methodology [186]. For the case of digital images-based visual inspection, this consists in using the entire dataset excluding one image to train the classifier that is later submitted to testing using the excluded image. The training process is repeated as many times as images are in the dataset, excluding a different image every time. To evaluate the performance of the detector, the metrics obtained for every iteration are averaged to obtain a global value.

The metrics used to evaluate the performance of the defect detectors are based on the matching between the classification output and the corresponding ground truth image. In this regard, four cases can be considered:

- true positive (TP), which occurs when the classifier successfully indicates the presence of a condition (e.g. corrosion), according to the ground truth;
- true negative (TN), which occurs when the classifier indicates that a condition is not fulfilled (e.g. non-defective pixel), and that coincides with the ground truth;
- false positive (FP), also known as false positive error or "false alarm", which occurs when the classifier indicates that a given condition has been fulfilled (e.g. presence of a crack), when actually it is not true, according to the ground truth; and
- false negative (FN), also known as false negative error, which occurs when the classifier indicates that a condition failed (e.g there is no crack), while actually the condition is met, according to the ground truth.

It is interesting to note that in many practical binary classification problems, the two classes are not symmetric, in the sense that the relevance of the two types of error is not the same. For example, in medical testing, false positives (detecting a disease when this is not present) and false negatives (not detecting a disease when it is present) are considered different error cases. It is common to take this fact into account to evaluate the performance of a binary classifier, obtaining ratios for the different types of errors, instead of total success ratios.

In our case, the defect detectors should be able to detect all the defective situations, despite this means that some false positive detections take place. On the contrary, a defect detector that does not provide false positive detections but which is not able to detect all the defective situations is considered a worse approach.

There exist several metrics and ratios to evaluate the performance of a binary classifier, and the election depends on the field of application [187]. Two of the most used metrics are the *True Positive Rate* (TPR) and the *False Positive Rate* (FPR). TPR, also known as sensitivity or recall, is the proportion of all positives that are successfully classified, while FPR, also known as fall-out, is the proportion of all negatives which are misclassified:

$$TPR = \frac{TP}{TP + FN}\,, \tag{4.1}$$

$$FPR = \frac{FP}{TN + FP}\,. \tag{4.2}$$

We use these two metrics to evaluate the performance of our defect detectors. In this regard, we make use of Receiver Operating Characteristic (ROC) curves. A ROC curve is a graphical plot of the TPR versus the FPR for a binary classification system, as its discrimination threshold is varied [129]. In a ROC curve, the ratios are usually provided in the [0, 1] range. An example is shown in Fig. 4.2. Notice that the best classifier would yield a point in the (0, 1) coordinate of the ROC space, corresponding to the upper left corner.

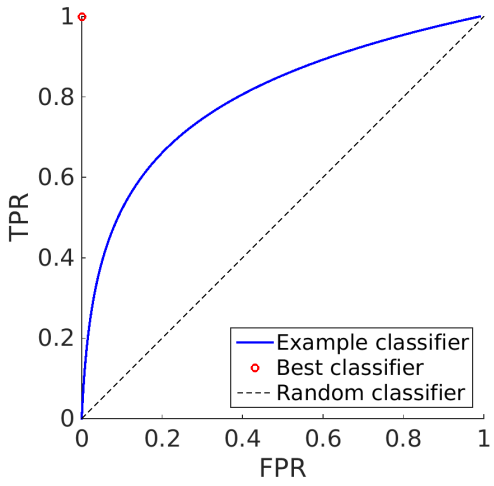


Figure 4.2: Example of a ROC curve.

The two extreme situations are produced when the classifier provides a point in the (1, 1) coordinate (upper right corner), what means that all the samples are classified as positive, and when it provides a performance corresponding to the (0, 0) coordinate, what indicates that all the samples are classified as negative. A completely random classifier gives a point along the diagonal line from the left bottom to the top right corner (see Fig. 4.2). Classifiers whose (FPR, TPR) point is below the main diagonal perform worse that a random classifier.

We also use ROC curves to set the appropriate value of the discriminant threshold. Given a ROC curve, we select the threshold value that corresponds to the point which is closest to the (0, 1) corner. If several points are situated at a similar distance, we select the one which corresponds to a higher TPR, despite this means also a higher FPR. This is so because we penalize more the false negative error than the false positive error.

To compare the ROC curves provided by different classifiers, we make use of the Area Under the Curve (AUC) [129]. The AUC is computed as the integral of the ROC curve, and hence takes values from 0 to 1. A classifier with a higher AUC performs better than a classifier with a lower AUC, so that an ideal classifier has an AUC of 1. On the other hand, an AUC of 0.5 can indicate random behaviour. The major advantage of the use of the AUC is that this is done independently of the decision criteria, thus eliminating the influence of the threshold value.

Another metrics that we use to evaluate the performance of our detectors is the *precision* (P). This indicates the proportion of positively classified samples which are actually positive:

$$P = \frac{TP}{TP + FP}\,. \tag{4.3}$$

This metrics is usually evaluated against the TPR (or recall) in a Precision-Recall (PR) curve. Like the ROC curves, a PR curve is created by plotting the precision against the

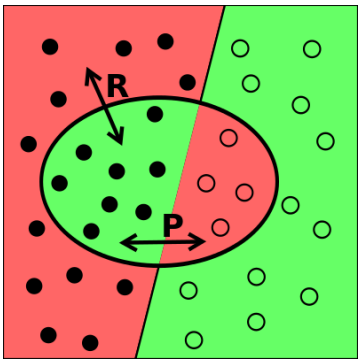


Figure 4.3: Precision and recall metrics. The positive items, according to the ground truth, are located to the left of the straight line, and the items retrieved as positive by the binary classifier are inside the oval. The red areas represent errors. Then, the red area located to the left, outside of the oval, represents the positive items that could not be retrieved (false negatives), while the red area inside the oval represents the items retrieved as positives that are not actually positives (false positives).

recall (R) obtained for different values of the discriminant threshold. In a PR curve the best performance corresponds to 100% of precision and 100% of recall (the upper right corner of the plot), which also yields the maximum AUC. Informally speaking, a high precision value means a low number of false positives, while a high recall value means a low number of false negatives. Figure 4.3 describes graphically the precision and recall metrics.

As mentioned before, to compute all these metrics, the classification output is compared with the ground truth image. Notice that pixels in the border between defective and non-defective areas can be wrongly labelled due to the inherent subjectivity during manual ground truth generation. In order to reduce the influence of this subjective process, a pixel classified as positive by a defect detector (e.g. corrosion) is considered in this work a true positive if there is a positive pixel in the ground truth image which is situated at five pixels or less.

Furthermore, we have taken special care with the crack detector. Cracks consist in a small collection of pixels which represents a little proportion of the total amount of pixels of the image. Because of that, the evaluation of the crack detector output in terms of pixels can provide results of difficult interpretation. Moreover, a successful detection of some pixels of a crack is considered enough to report the existence of this defective situation. For these reasons, we consider cracks as entities, so that it is enough that the defect detector identifies part of the crack. Therefore, using this methodology, every image in the dataset comprises one or more entities (cracks), and the rest of pixels is considered the background. Notice that, the FPR can not be computed in this case since the true negatives can not be accounted for. All crack detectors have thus been assessed using precision/recall metrics.

The execution time has also been measured for the different defect detectors. The values provided correspond to a desktop computer fitted with an Intel Core i7-5820K processor at 3.30 GHz and 32 GB of RAM, which runs GNU/Linux Ubuntu.

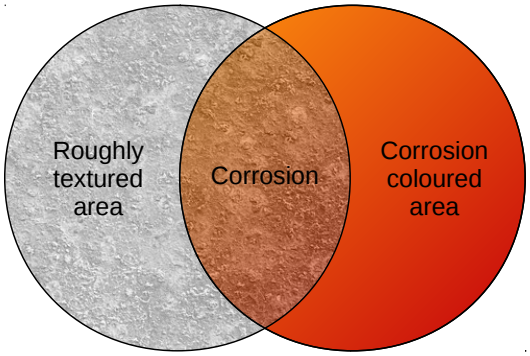


Figure 4.4: Venn diagram for corrosion definition.

4.3 Detection of Corrosion on Vessel Structures

4.3.1 General Overview

This section describes the development of a corrosion detector based on a cascade of classifiers, whose different stages are *weak classifiers*. The idea is to chain different fast classifiers with poor performance in order to obtain a global classifier attaining a much better global performance. To this end, each weak classifier takes profit from different features of the items to classify, reducing the number of false positive detections at each stage. For a good global performance, the classifiers must exhibit a reduced false negative rate.

Two features are considered to describe corrosion: texture and colour. Moreover, in our case, the corrosion detector comprises a two-stage cascade, one stage for each feature. One of these stages is based on the premise that corroded areas present a rough texture, while the other stage checks whether the inspected area presents a colour typically observed in corroded surfaces. This stage is based on machine learning and entails a previous training process to learn the colour of corrosion. Figure 4.4 depicts the corrosion definition used in our approach.

The following sections, from 4.3.2 to 4.3.5, discuss on different features that have been considered to describe corrosion colour and texture. After that, Section 4.3.6 presents the main approach which combines the aforementioned features to build up the corrosion detector. In this section, an alternative approach is also introduced, and its performance is compared with the main approach. Finally, Section 4.3.7 draws some conclusions regarding the corrosion detection methods.

4.3.2 Modelling Corrosion Colour through Global Colour Maps

To describe the colour of corrosion we firstly propose the use of a global histogram modelling the distribution of colour in corroded areas. This is computed during a learning stage where all the corroded pixels from the input images add one vote to the corresponding bin of the histogram. Two different colour spaces have been considered. On the one hand, we have

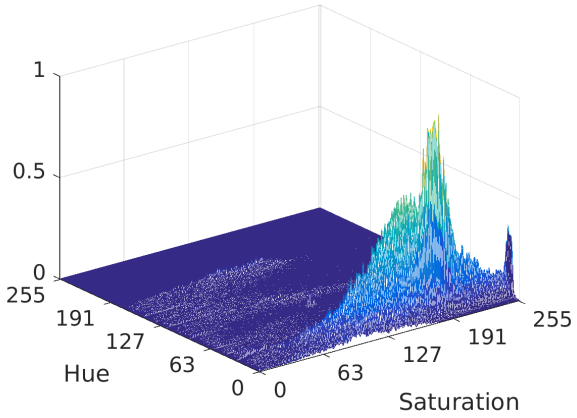


Figure 4.5: Hue-Saturation histogram for corroded pixels.

contemplated the RGB space because this colour model is the most common regarding sensing, representation and display of images through electronic systems. To implement the RGB histogram, a three-dimensional structure is used, where each dimension represents one colour channel and spans the corresponding 256 intensity levels.

On the other hand, we also contemplate the HSV colour model in order to separate colour and intensity information. Indeed, the intensity information is not used in our approach, what allows learning the corrosion colour disregarding the amount of light that illuminates the surface. Therefore, the histogram can be implemented using a two-dimensional structure to account for all the combinations of the pair hue-saturation. By way of example, Fig. 4.5 shows the normalized HS histogram generated from all the corroded pixels of the image dataset. As can be observed, most of the colours that appear in corroded areas are confined in a bounded subspace of the HS plane, which corresponds to reddish colours.

Next, for both colour spaces, the corresponding histogram is thresholded in order to reduce the influence of colours with low presence in corroded areas, resulting in a colour map for each colour space. During this process, all the colours corresponding to bins which account less than τ_C pixels are discarded as corrosion colours by setting to 0 the corresponding map entry; the other colours are recorded as 1 in the map.

A last step fills the gaps of the map that may arise among the corrosion area and, thus, increase the generalization of the histogram. This process consists in the morphological operation *closing* applied to the colour map. This operator firstly dilates the corrosion colour area using a structuring element to fill the inner gaps, and then erodes the area using the same structuring element to recover its original size.

Summing up, the corrosion colour is modelled through a colour map which actually corresponds to one or more clouds in a 2/3-dimensional colour space, collecting the colours that typically appear in corroded surfaces. The pseudocode for the corrosion colour global histogram computation is provided as Alg. 4.1. This pseudocode corresponds to the computation

Algorithm 4.1 Procedure for HS global histogram computation.

procedure CORROSION_COLOUR_GLOBAL_HISTOGRAM_COMPUTATION(I, GT, τ_C)
    I: input image set
    GT: image set comprising a ground-truth image for each image in I
    τ_C: threshold to binarize the colour histogram
    Initialize the histogram $\mathcal{HS}$ with zeros
    for all image *img* in I **do**
        for all pixel p of *img* **do**
            if p is labelled as *corrosion* in the ground truth image **then**
                Get the hue (h) and saturation (s) values of p
                Increase $\mathcal{HS}(h, s)$ one unit
            end if
        end for
    end for
    Build the $\mathcal{HS}$ map for corrosion using threshold τ_C
    Closing the $\mathcal{HS}$ map
    Save $\mathcal{HS}$ to a file
end procedure

of the HS histogram; minor modifications are required to compute the RGB histogram.

4.3.3 Modelling Corrosion Colour through Local Stacked Histograms

The second corrosion colour model that we propose consists in learning the colours that appear in the neighbourhoods of corroded pixels. Given a corroded pixel, the histogram for each colour channel is computed for its $N \times N$ surrounding pixels. The resulting histograms are stacked together to perform what we call a *codeword*. As for the previous model of corrosion colour, RGB and HS colour spaces have been considered, so that codewords result from stacking three (R-G-B) or two (H-S) histograms together. These histograms are downsampled from 256 to 32 levels in order to reduce its dimensionality and sensitivity to noise. Therefore, each codeword comprises 96 values, if the RGB colour space is used, and 64, if HS is employed.

By way of example, Fig. 4.6 shows the codeword corresponding to a corroded pixel when the RGB colour space is used. As can be observed, this codeword does not preserve the spatial arrangement of intensity levels nor the relationship between colour channels for the same pixel.

When using this model, the learning stage consists in computing the codewords corresponding to the corroded pixels in the image dataset and clustering them by means of the well-known K-means algorithm [188]. The clustering process is performed in order to make the dictionary more compact and general. The resulting K codeword models represent the information learned about the colour distribution around corroded pixels. Algorithm 4.2 presents the pseudocode for the codeword dictionary computation when the RGB colour space is used.

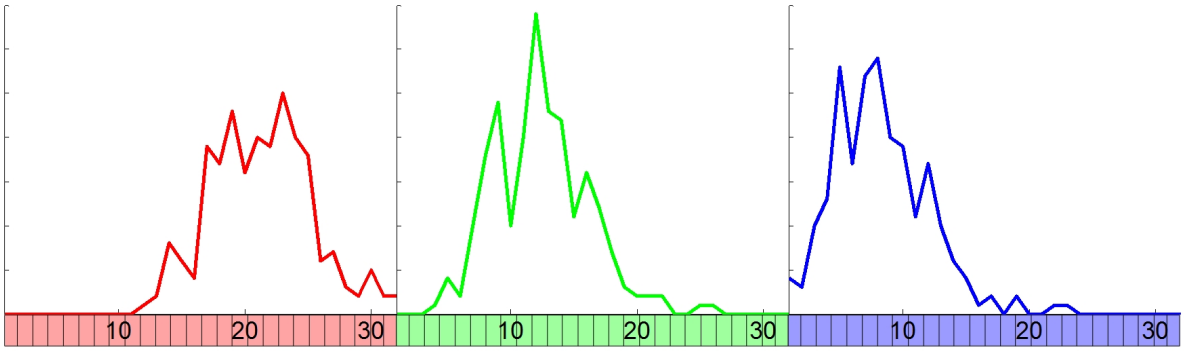


Figure 4.6: Codeword including colour information used to describe corrosion.

4.3.4 Modelling Corrosion Texture by means of GLCM Energy

To describe the texture of corrosion we propose using the symmetric Gray-Level Co-occurrence Matrix (GLCM). This matrix is defined over an image to be the distribution of co-occurring pixel values (gray-scale values, or colours) at a given offset. That is, every (i, j) value of the GLCM indicates the number of times that i and j pixel values occur in the input image at a distance given by the offset.

To evaluate the roughness of the image, we compute the *energy*, or Angular Second Moment (ASM), of the GLCM [188]. This is computed by means of

$$E = \sum_i \sum_j p(i,j)^2, \tag{4.4}$$

where $p(i, j)$ is the probability of the occurrence of values i and j at the specified offset. The energy of a texture is related to its roughness so that the higher is the roughness the lower is the energy.

In our approach, the GLCM is calculated for downsampled intensity values between 0 and 31, for a given distance of d pixels and a direction $\alpha \in k\pi/4$ rad, where $k \in [0, 7]$, so that eight directions are considered in order to compute an isotropic energy.

4.3.5 Modelling Corrosion Texture by means of Law's Filters Responses

Law's filters responses [189, 190] are also considered for describing corrosion texture. These filters are used to enhance different features of every material texture by means of convolution. For example, the following 1D five-component basic filters can be used to detect different features:

level L5 = [1 4 6 4 1] edge E5 = [-1 -2 0 2 1]
spot S5 = [-1 0 2 0 -1] wave W5 = [-1 2 0 -2 1]
ripple R5 = [1 -4 6 -4 1]

Algorithm 4.2 Procedure for RGB codewords dictionary generation.

```
1: procedure CODEWORDS_DICTIONARY_GENERATOR(I, GT, N, K)
2:     I: input image set
3:     GT: image set comprising a ground-truth image for each image in I
4:     N: patch size in pixels
5:     K: Number of codewords to generate
6:     for all image img in I do
7:         for all pixel p of img do
8:             if p is labelled as corrosion in the ground truth image then
9:                 Get the N×N patch Π centred at p
10:                Compute the 32-bin histogram for the red channel of Π
11:                Compute the 32-bin histogram for the green channel of Π
12:                Compute the 32-bin histogram for the blue channel of Π
13:                Stack the three histograms together
14:                Save the resulting codeword
15:            end if
16:        end for
17:    end for
18:    Cluster codewords to K models using K-means
19:    Save the models to a file
20: end procedure
```

To describe a texture, the corresponding gray-level patch is convolved with the set of energy filters and different statistical measures are taken over an $N \times N$ neighbourhood of the filter responses, which finally constitute the texture descriptor. The statistical measures used are the standard deviation, the mean of positive values and the mean of negative values, which are respectively computed as

$$\sigma = \sqrt{\sum_{u=0,v=0}^{N,N} \frac{(c(u,v)-\mu)^2}{N^2}}\,, \tag{4.5}$$

$$\mu^+ = \frac{\sum_{u,v|c(u,v)>0}^{N,N} c(u,v)}{N^2}\,, \tag{4.6}$$

$$\mu^- = \frac{\sum_{u,v|c(u,v)<0}^{N,N} c(u,v)}{N^2}\,, \tag{4.7}$$

where c is the output of the convolution, u and v are used to iterate over the $N \times N$ neighbourhood, and

$$\mu = \frac{1}{N^2} \sum c(u,v). \tag{4.8}$$

A filter bank has been generated in order to characterize the corrosion texture through different scales and regardless of the orientation. This comprises 75 2D filters that have been

computed as indicated in the following steps:

1. Each five-component 1D basic filter has been operated to obtain the corresponding nine-component filter. By way of example, L9 = L5 $\otimes$ L5 = [1 8 28 56 70 56 28 8 1], where $\otimes$ is the polynomials multiplication operation.

2. Each nine-component filter has been operated to obtain the corresponding seventeen-component filter.

3. Each two filters with the same number of components have been combined to obtain the 2D filters. For example, combining L5 with E5, the following 5×5 filter is obtained:

$$\begin{bmatrix}1\\4\\6\\4\\1\end{bmatrix}\begin{bmatrix}-1 & -2 & 0 & 2 & 1\end{bmatrix} = \begin{bmatrix}-1 & -2 & 0 & 2 & 1\\ -4 & -8 & 0 & 8 & 4\\ -6 & -12 & 0 & 12 & 6\\ -4 & -8 & 0 & 8 & 4\\ -1 & -2 & 0 & 2 & 1\end{bmatrix}. \tag{4.9}$$

   Therefore, 25 2D filters are obtained for each length considered (5, 9 and 17), resulting into 75 2D filters.

4. Each 2D filter is modified to make it sum be 0 and to be of unit gain. To do that, we subtract the mean value of the filter and divide by the sum of its positive elements.

By way of example, Fig. 4.7 shows some of the resulting filters used to describe the corrosion texture. These filters are convolved with the gray-scale image to obtain the filter output, which is then used to compute the statistical measures.

Notice that, given two 1D row filters $F1$ and $F2$, the combination $F1'F2$ provides the same 2D filter as $F2'F1$, but rotated 90 degrees. We take advantage of this property to make the texture features invariant to orientation. To do that, the output obtained using each operation $F1'F2$ is averaged with the output obtained with $F2'F1$. Doing this, the 75 filter outputs are compacted to 45.

After combining the outputs of rotated filters, the final vector describing the texture of a given $N \times N$ patch amounts to 135 components (45 filters responses × 3 statistical measures).

4.3.6 Classifier Design

As anticipated before, to implement the classifier we have combined a corrosion colour model with a corrosion texture model among the ones presented in the previous sections. These models are used to classify every pixel in the image as potentially corroded or not.

In the following sections, we focus on two approaches. Section 4.3.6.1 introduces our main approach, which consists in using the texture model based on the GLCM energy in

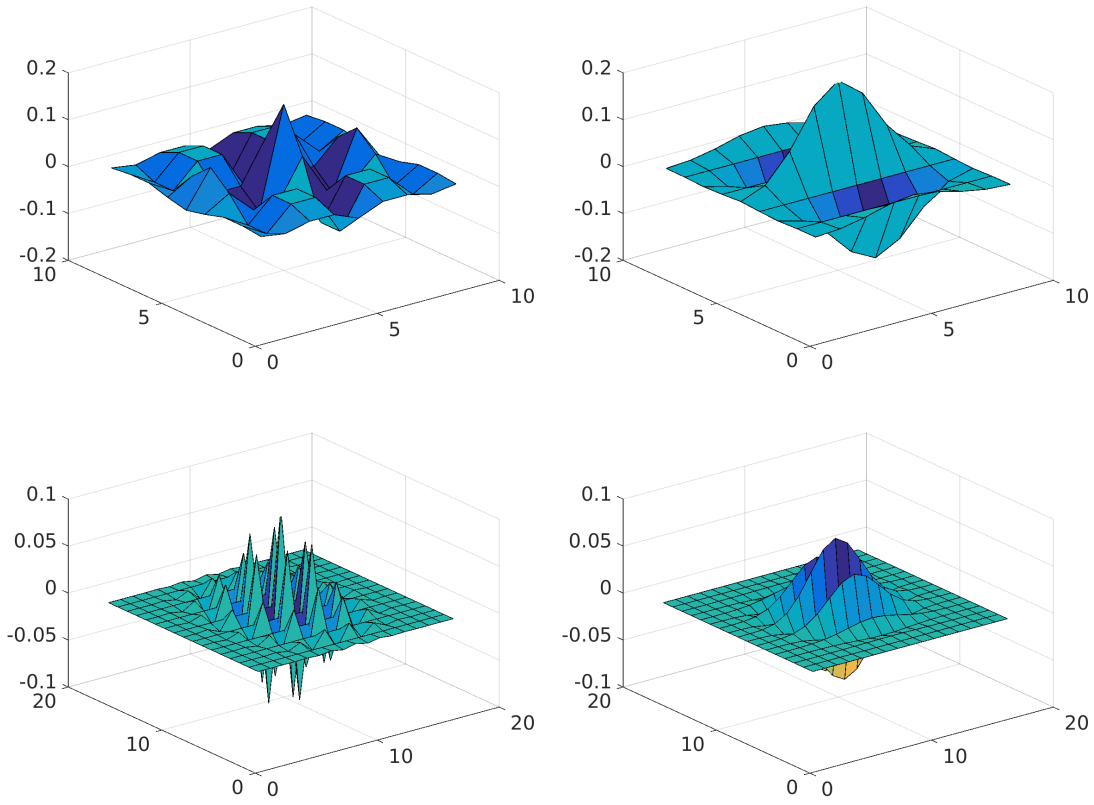


Figure 4.7: Some 2D Law's filters used to describe corrosion texture.

combination with a corrosion colour model. The feature used in this texture model is a clear indicator of the surface roughness regardless of the specific spatial distribution of intensities. The two different colour models involved in Sections 4.3.2 and 4.3.3 are considered, and both are evaluated.

Section 4.3.6.2 describes an alternative approach. This is based on using the Law's filters responses texture model, to account for the intensity distribution that appears in the proximity of corroded pixels. The idea is to evaluate whether the corrosion texture follows some specific pattern that can be learned or, on the contrary, it is just a matter of roughness. The resulting texture stage is used together with the corrosion colour model which has presented the best performance during the main approach evaluation.

4.3.6.1 Main Approach

As previously said, our main approach for corrosion detection consists in using the GLCM energy-based texture model in combination with a corrosion colour model. The order of application of the stages depends on its capability for discarding non-defective areas, together with the computational cost of each stage. Therefore, when the global colour maps are used,

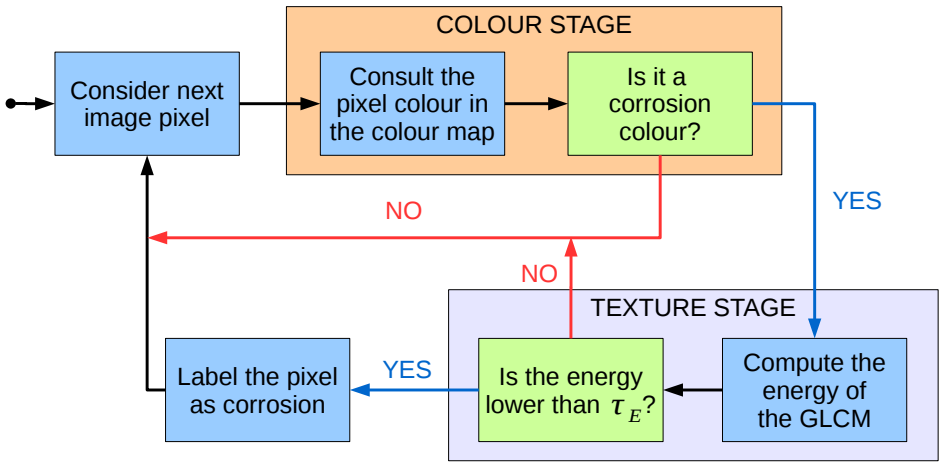


Figure 4.8: Flowchart of the corrosion detector using a global colour map to implement the colour stage.

the colour model is executed in first place since this just entails a query in the colour map, while the energy stage requires computing the GLCM for every pixel of the image. Nevertheless, when this texture model is combined with the colour stage based on local stacked histograms (codewords), the texture stage is executed in first place in order to reduce the number of queries in the codewords dictionary.

Regarding the texture stage, this classifies every pixel as potentially corroded or not depending on whether the energy of the GLCM computed for the surrounding $N \times N$ patch is below a given threshold τ_E. Remember than the lower the energy of a texture, the higher its roughness.

Regarding the colour stage:

- When the global histogram corrosion colour model is used to implement the colour stage, this consists in a simple query to the previously computed colour map to check whether the colour of the pixel is common in corroded areas. The flowchart of the entire corrosion detector when this colour model is used can be found in Fig. 4.8, while the pseudocode for the classification procedure, for the case of using the HS histogram, is provided as Alg. 4.3.

- When the corrosion colour model based on stacked local histograms is used to implement the colour stage, this consists in building the codeword for the $N \times N$ image patch centred in the current pixel and comparing with the models of the dictionary by means of the Bhattacharyya distance

$$D_B = -\log(BC), \tag{4.10}$$

Algorithm 4.3 Corrosion detector using the colour model based on a global colour map.

```
1: procedure CORROSION_DETECTOR_I(img, N, τ_E)
2:     img: input colour image
3:     N: patch size in pixels
4:     τ_E: energy threshold
5:     Load the hue-saturation colour map HS
6:     Initialize img_out to no corrosion
7:     for all pixel p in img do
8:         Get the hue (h) and saturation (s) of p
9:         if HS(h, s) > 0 then                    ▷ Proceed with the texture stage
10:            Get the N×N patch Π centred at p
11:            Compute the GLCM energy e of Π
12:            if e < τ_E then
13:                Label p as corrosion in img_out
14:            end if
15:        end if
16:    end for
17:    return img_out
18: end procedure
```

with the Bhattacharyya coefficient BC given by

$$BC = \sum_{x \in \mathcal{X}} \sqrt{p_c(x) p_m(x)}\,, \tag{4.11}$$

where $\mathcal{X}$ refers to the histograms domain and p_c and p_m are histograms from, respectively, the codeword and the model from the dictionary.

A pixel is labelled as corroded as soon as a model is found in the dictionary such that $D_B < \tau_D$. Therefore, the approach does not intend to determine which is the closest model, but whether the patch is close enough to any model of corrosion. As a consequence, an important reduction in the computation time is obtained.

The flowchart for the complete algorithm using this colour model is shown in Fig. 4.9 and its pseudocode, for the case of using the RGB colour space, can be found as Alg. 4.4.

We have assessed the corrosion detection performance of the four combinations, that is to say, using the two colour models and the two colour spaces considered. To do that, different parameters have been used trying to find the best configuration for each combination. Regarding the texture stage, the GLCM has been computed using a distance $d = 5$ pixels and an image patch of 15×15 pixels, while the energy threshold τ_E has been set to different values covering the full [0, 1] range.

Concerning the colour stage based on the global histogram colour model, the configuration of its parameters depends on the colour space used. When the RGB colour space is utilized, the binarization threshold τ_C is set to three pixels, and the structuring element used for the

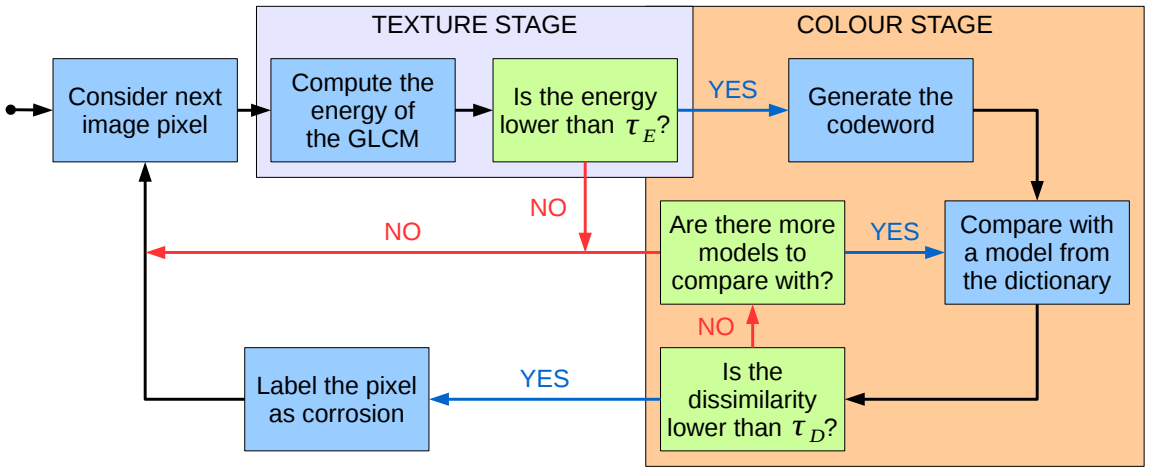


Figure 4.9: Flowchart of the corrosion detector using the local stacked histograms colour model to implement the colour stage.

Algorithm 4.4 Corrosion detector using the local stacked histograms colour model.

procedure CORROSION_DETECTOR_II(img, N, τ_E, τ_D)
    img: input colour image
    N: patch size in pixels
    τ_E, τ_D: energy and dissimilarity thresholds
    Load codewords dictionary
    Initialize img_out to *no corrosion*
    for all pixel p in img **do**
        Get the $N \times N$ patch Π centred at p
        Compute the energy e of Π
        if $e < \tau_E$ **then** ▷ Proceed with the colour stage
            Compute the 32-bin histogram for the red channel of Π
            Compute the 32-bin histogram for the green channel of Π
            Compute the 32-bin histogram for the blue channel of Π
            Stack the three histograms together into a codeword CW
            while there are more models mod in the dictionary **and**
                p has not been labelled in img_out **do**
                Calculate the Bhattacharyya distance D between CW and the model mod
                if $D < \tau_D$ **then** ▷ The codeword is similar to the model
                    Label p as *corrosion* in img_out
                end if
            end while
        end if
    end for
    return img_out
end procedure

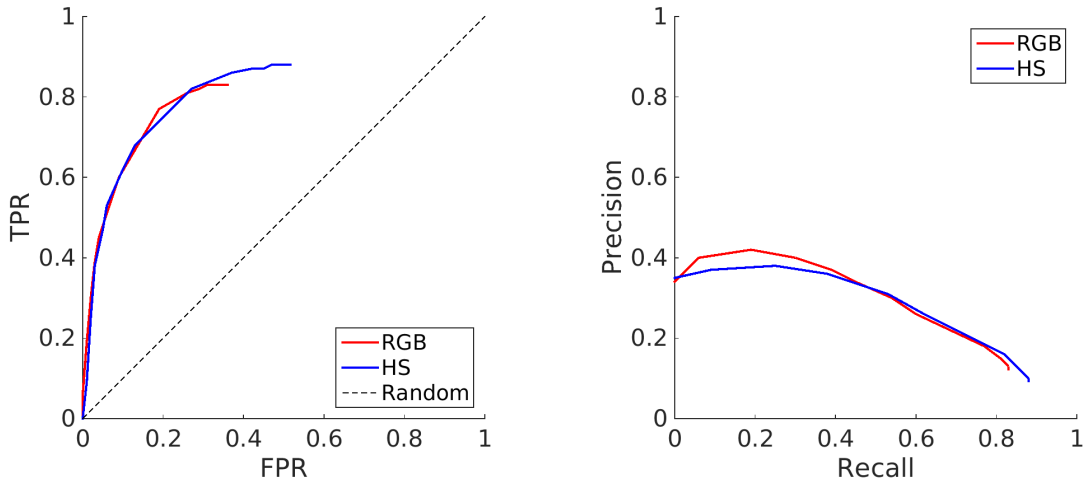


Figure 4.10: Performance of the corrosion detector using the global histogram colour model: (left) ROC curves and (right) PR curves. Each curve corresponds to the use of a different colour space, and has been generated by variation of the threshold τ_E.

closing morphological operator consists in a cube whose edges are five pixels in length. When the colour map is computed for the HS colour space, τ_C is set to ten pixels, and the structuring element consists in a circle of five-pixel radius.

Regarding the local stacked histograms colour model, its parameters are configured regardless of the selected colour space. Local histograms are computed for 15×15-pixel patches, and to create the codewords dictionary, three different sizes have been evaluated: 100, 300 and 500 codeword models. Notice that if the dictionary contains too many codewords, the classification process may be affected by overfitting. Furthermore, when using a larger dictionary, more comparisons are performed prior to discarding non-defective pixels, what means a longer processing time. Finally, both the energy and the dissimilarity thresholds, i.e. τ_E and τ_D respectively, have been evaluated for different values in the [0, 1] range.

Figure 4.10 shows the ROC and PR curves obtained for the corrosion detector when global colour maps are used. The plots compare the detection performance when using the RGB and HS colour maps, for different values of the energy threshold τ_E. On the one hand, the ROC curves are very similar, laying at approximately the same distance to the (0,1) corner, i.e. the perfect classifier. On the other hand, the use of the RGB colour space allows achieving a bit higher precision at low recall values, while precision is almost the same for higher recall values.

The same performance metrics have been computed for the case of using the colour model based on local stacked histograms, and are provided in Fig. 4.11. Each ROC/PR curve in the figure corresponds to the use of a specific dictionary size (100, 300 and 500) and energy threshold τ_E (0.2, 0.4 and 0.6), and has been generated by variation of the codeword dissimilarity threshold τ_D. To facilitate the comparison of the ROC curves, Table 4.1 provides

Table 4.1: AUC values for the corrosion detector using the local stacked histograms colour model. These values correspond to the ROC curves of Fig. 4.11 [left]. The best results are highlighted in blue.

			Threshold τ_E		
			0.2	**0.4**	**0.6**
Codewords dictionary	**100 models**	**RGB**	0.877	0.892	0.891
		HS	0.875	0.879	0.875
	300 models	**RGB**	0.867	0.880	0.883
		HS	0.860	0.858	0.850
	500 models	**RGB**	0.863	0.871	0.870
		HS	0.859	0.852	0.842

the corresponding AUC values.

As can be observed, all the combinations provide a similar performance, presenting AUC values between 0.84 and 0.89. The use of RGB codewords always provides slightly better AUC values than using HS codewords, despite the difference is below 2% in most cases. Regarding the size of the dictionary, the best results are obtained when using the smaller dictionary, which contains just 100 codewords. That means that the use of larger dictionaries, with 300, 500 or more codewords, seems to lead to overfitting and loss of generalization.

Similarly to the other colour model, the use of RGB codewords allows achieving a higher precision at low recall values, while, for higher recall values, HS and RGB codewords provide a similar precision.

Table 4.2 provides the execution times required by the different configurations of the corrosion detector. These values correspond to the mean execution times per pixel using the parameter configuration that provides the best performance, based on the ROC curves presented in Fig. 4.10 [left] and Fig. 4.11 [left]. On the basis of these values, the table also provides the expected execution times that would be required to process a 1024×768-pixel image. Notice that these are just estimations computed averaging the observed values. Actually, the processing time depends on the percentage of corroded area in the specific image considered.

As can be observed, the use of the global colour map model in the colour stage clearly makes the algorithm much faster. This is because the query in the global histogram entails much less time than the search in the codewords dictionary. Furthermore, when using the colour stage based on the local stacked histograms, the use of RGB codewords entails an increment of 40% in the execution time, regarding the use of HS codewords, as could be expected.

Figure 4.12 shows some results of the corrosion detector using the different colour models and colour spaces. As can be observed, the different configurations are able to successfully detect the corroded areas in the input images, and their results approximately match the ground truth.

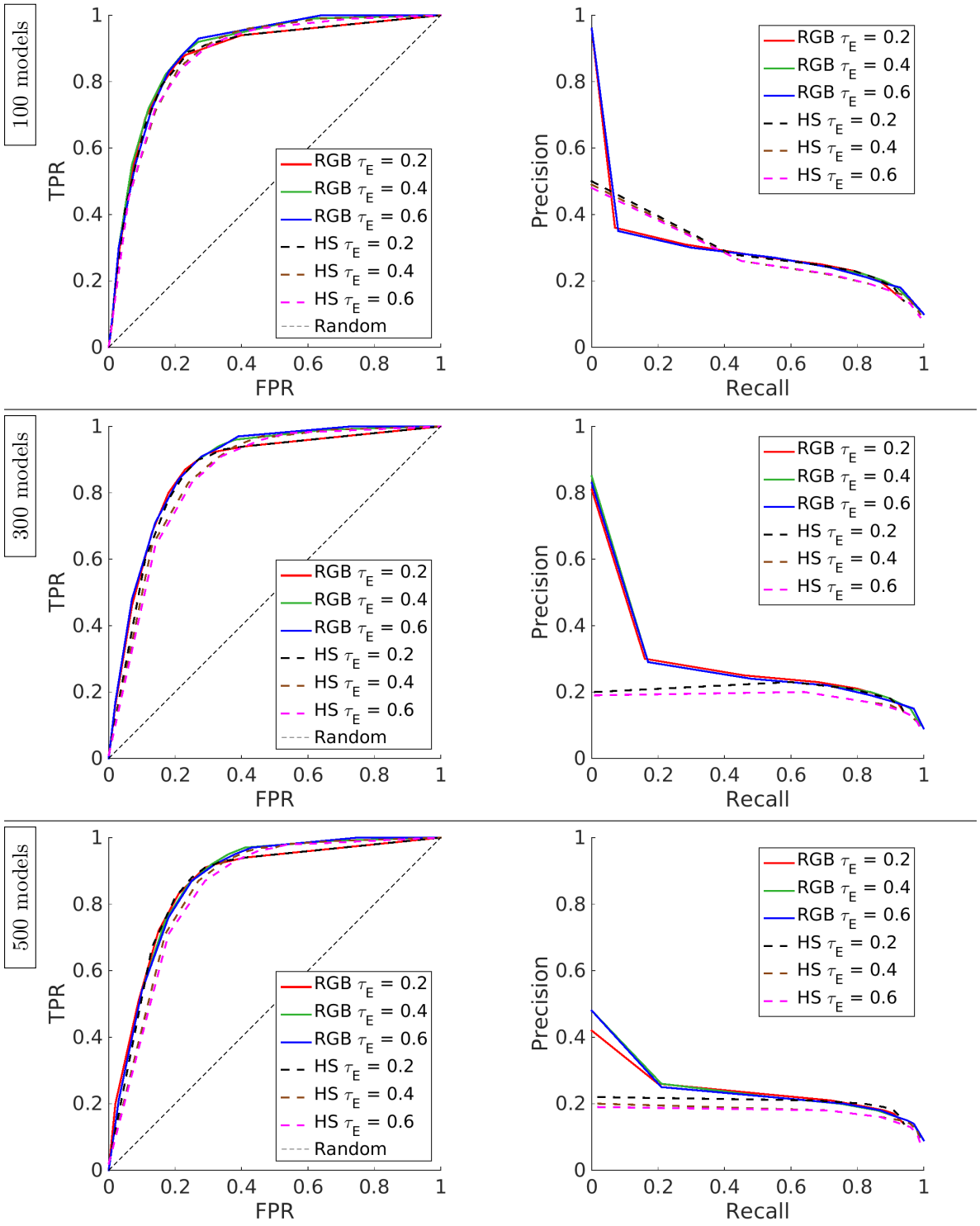


Figure 4.11: Performance of the corrosion detector using the local stacked histograms colour model: (left) ROC curves and (right) PR curves. Each curve corresponds to the use of a different combination colour space/energy threshold, and has been generated by variation of the threshold τ_D. Each row corresponds to a different dictionary size.

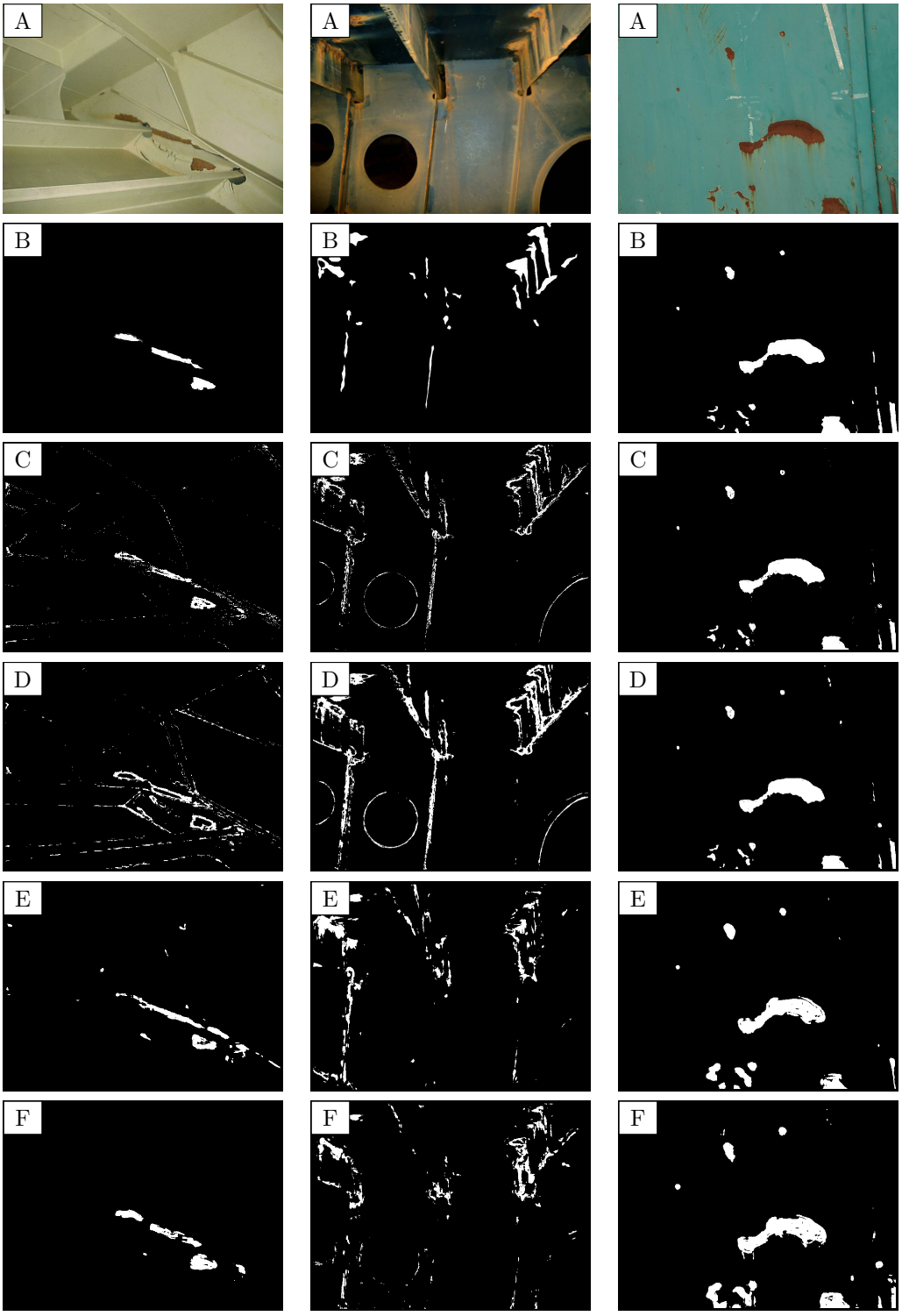


Figure 4.12: Corrosion detection results for some images of the dataset: (A) input image, (B) ground truth, (C-D) corrosion detection outputs using, respectively, RGB and HS global colour maps, (E-F) corrosion detection outputs using, respectively, RGB and HS codewords.

Table 4.2: Execution times of the corrosion detector using the different colour models. Mean values computed using the parameter configurations that provide the best performance.

	Colour map		**Local histograms**	
	RGB	**HS**	**RGB**	**HS**
μs/pixel	2.00	2.96	31.12	22.26
s/image*	1.57	2.33	24.47	17.50

* for a 1024×768 image.

In general terms, the use of the proposed colour models, in combination with the GLCM energy stage, provides good classification results. On the one hand, the use of the local stacked histograms for describing colour allows achieving higher true positive rates with less false positive detections. On the other hand, global colour maps lead to pretty good results at a speed of about ten times faster.

Parameter configuration

By way of summary, the following lines discuss about the configuration of the different parameters involved in the main approach for corrosion detection and that have been already introduced somewhere in the previous pages. Regarding the texture stage based on the GLCM energy:

- The patch size is configured to 15×15 pixels. This size has been set to perceive the roughness/homogeneity of the texture. Notice that using a too small patch would make difficult a successful perception of the texture, while a too large patch would increase the computation time and would reduce the detection accuracy.

- The GLCM is computed for the image downsampled to 32 intensity values in order to reduce the effect of noise.

- The GLCM is computed using a distance $d = 5$ pixels and 8 different directions. This offset has been set considering the texture of corrosion and the size of the patch. The use of 8 directions allows obtaining an isotropic measure (i.e. independent of orientation).

- The energy threshold τ_E is the discriminant parameter, and has been set to different values covering the [0, 1] range.

The parameters of the colour stage based on the use of a global map have been configured as follows:

- The threshold τ_C is used to discriminate the colors that are considered common in corroded areas from those that do not tend to appear in such defective surfaces. When

using the RGB color space, this threshold has been set to 3, while this has been configured to 10 when the HS map is being computed. This latter value is higher than the first one because the cloud of corrosion colours is more compact in HS colour space, while the dispersion of the same colours represented in the RGB histogram is higher.

- Similarly to the previous point, the closing operator when using the RGB color space consists in a cube whose edges are five pixels in length, while this consists in a circle of five-pixel radius when the HS colour space is employed.

Finally, the parameters of the colour stage based on the use of local stacked histograms have been configured as follows:

- The patch size has been set to 15×15 pixels to use the same patches considered in the texture-based stage.
- The colour channels are downsampled to 32 values in order to reduce the effect of noise in the image (for the same reason as before).
- The dictionary size has been set to 100, 300 and 500 codewords. A dictionary with just 100 models has been considered as a reduced dictionary. The dictionaries with 300 and 500 models are evaluated to check whether the use of more codewords improves the detection performance or, on the contrary, leads to overfitting.
- The dissimilarity threshold τ_D is the discriminant parameter in this case, and has been set to different values covering the [0, 1] range.

4.3.6.2 Alternative Approach

The alternative approach consists in using the Law's filters responses corrosion texture model, introduced in Section 4.3.5, to learn the specific texture of the corroded areas. The idea is to check whether corroded surfaces present some specific texture which can be learned. Therefore, a new texture stage based on supervised learning has been designed for that purpose.

This stage has been combined with the colour stage which makes use of HS local stacked histograms, since this colour stage has provided the best results during the performance assessment reported in the previous section. Besides, the selection of the HS colour channels, instead of RGB, is due to the reduction in the execution time observed when using codewords for these two channels, while its detection ratios are almost as good as the ones obtained when using RGB codewords. The codeword dictionary used contains 100 corrosion colour models.

The new texture stage makes use of *Adaptive Boosting*, also known as AdaBoost, for both learning and classifying corroded areas. This is a machine learning algorithm first introduced by Freund and Schapire [146] for constructing a strong classifier as a linear combination of

simple weak classifiers or features $h_t(x)$:

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x). \tag{4.12}$$

The strong or final classifier $H_t(x)$ is defined as

$$H(x) = sign(f(x)). \tag{4.13}$$

AdaBoost is adaptive in the sense that each stage tries to select the feature that best classifies all the samples, giving more importance to the ones misclassified by the previous stage. AdaBoost can thus be seen as a feature selector.

We propose a new texture stage making use of AdaBoost for both learning and classifying. To this end, AdaBoost has been fed with the statistical measures obtained after convolving Law's texture energy filters with patches centred at both corroded and non-corroded pixels. The intention is to enforce the learning of the features that allow to differentiate the two kinds of surface. The patches have been defined as square areas of 15×15 pixels.

During the learning process, AdaBoost has been fed with feature vectors of both classes. A set of 300 pixels from each class have been randomly selected from all the images of the training dataset. The surrounding 15×15 patches centred at the selected pixels have been convolved with Law's filters and the aforementioned statistical measures have been computed. One half of the samples has been used to train AdaBoost, while the other half has been used as control samples to assess the performance of the resulting classifiers. After this process, the output obtained is a set of weak classifiers together with their weights, that is supposed to correctly separate those samples belonging to corroded areas from those which are not.

We have used the AdaBoost implementation found in the GML AdaBoost Matlab Toolbox, implemented by Alexander Vezhnevets from the MSU Graphics and Media Lab[1]. This implementation makes use of *Classification and Regression Trees* (CART) as weak classifiers. A CART is a binary tree graph used for classification tasks, where leaves represent the classification result and nodes represent some predicate. A classification process consists in traversing the tree and selecting at each node whether we should proceed through the left or right branch, depending on the value of a predicate. Finally, when a leaf is reached, the sample is classified in accordance to the associated value. An example of CART can be found in Fig. 4.13.

Let $S = [(x^1, y^1), ..., (x^m, y^m)]$ be a sequence of training examples, where x^i belongs to the domain space $\mathcal{X} \in R^n$ (real valued vector with dimensionality n, $x^i = (x_1^i, ..., x_n^i)$), and each label y^i belongs to $\mathcal{Y} = \{-1, 1\}$. The AdaBoost implementation used constructs every node of the CART as follows:

1. for all the n dimensions, find the threshold Θ_n that separates S with least error,

[1] http://graphics.cs.msu.ru/en

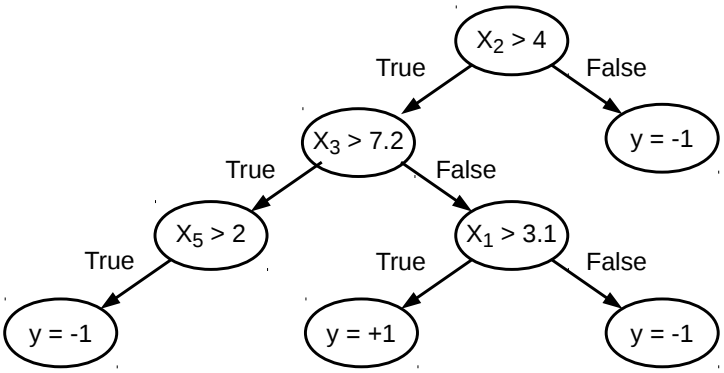


Figure 4.13: Example of CART.

2. choose the dimension j with least error, and incorporate a node with predicate $x_j > \Theta_j$,
3. connect branches *true* and *false* to leafs that have the respective classification.

Let *error of leaf* be the probability of a sample being misclassified if, during the tree traverse, we stop at this leaf. To construct the whole tree the following steps are used:

1. add the root node,
2. choose the leaf with the largest error,
3. add a node using just those training samples that are associated with the chosen leaf,
4. replace the chosen leaf by the added node,
5. repeat steps 2-4 until all leaves have zero error or a predefined number of steps have been performed.

Once AdaBoost has been trained, the resulting classifier can be included in our corrosion detector. The flow diagram for this alternative approach, including the colour stage, is shown in Figure 4.14, while the corresponding pseudocode is provided in Alg. 4.5. As can be observed, all the convolutions using the Law's filters are performed once at the beginning of the process, in order to avoid repeating computations.

We have considered three versions of AdaBoost: *Real*, *Gentle* and *Modest*. *Real AdaBoost* is the generalization of a basic AdaBoost algorithm. *Gentle AdaBoost* [191] is a more robust and stable version of Real AdaBoost, used, for example, by the well-known Viola-Jones face detector [192]. Finally, *Modest AdaBoost* [193] is a version mostly aimed for better resistance to overfitting. These three versions are available in the AdaBoost toolbox we have used.

The AdaBoost parameters have been configured as follows:

- the maximum number of boosting iterations, i.e. the number of weak classifiers that make up the final classifier, has been set to 100.

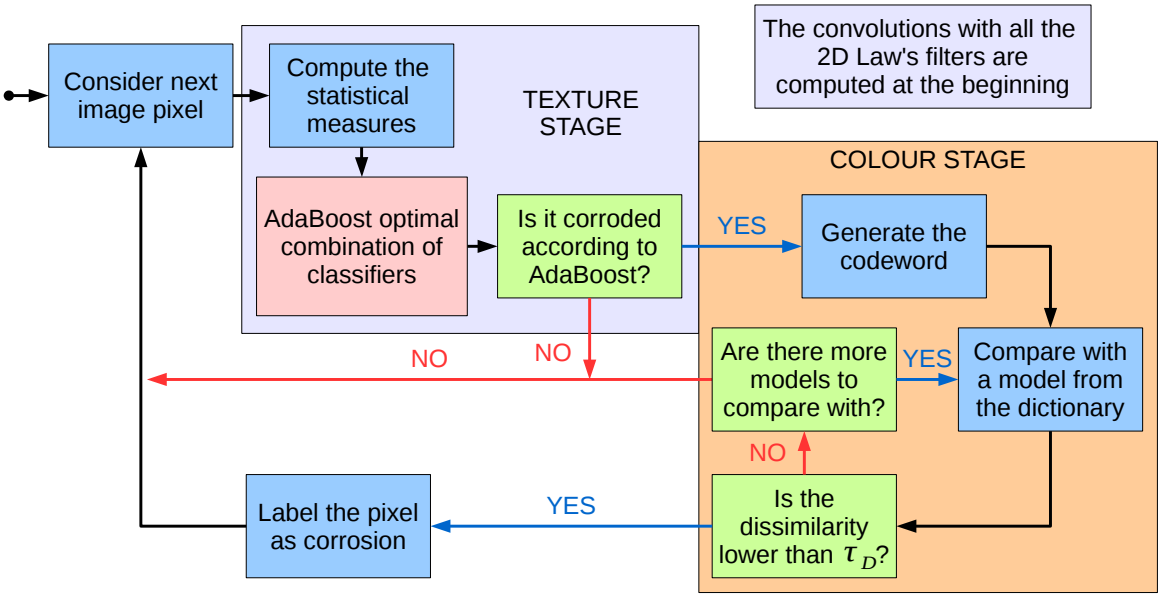


Figure 4.14: Flowchart of the corrosion detector using AdaBoost and Law's filters responses to describe texture.

- the depth of the CART, which determines how good the weak classifiers are, has been set to three levels.

Both parameters have been configured to improve the detection performance without prolonging the learning time unnecessarily. By way of example, the classification errors obtained after the training process using samples from all the images in the dataset have been: 15,8% for both Real and Gentle methods, and 18,8% for the Modest version. Notice that these error percentages include both false positive and false negative detections.

Figure 4.15 shows the ROC and PR curves corresponding to the three versions of AdaBoost. These have been obtained by variation of the τ_D threshold of the colour stage. As can be observed, the Modest version clearly outperforms both the Real and the Gentle implementations, which provide both very similar results. This is probably due to the generalization capability and resistance to overfitting of this AdaBoost version.

The mean execution time for the dataset using this alternative approach is about 54.3 μs/pixel, what supposes an average of 42.69 s/image when considering images of 1024×768 pixels. The increment in time with regard to the use of the energy-based texture model is clearly due to the computation of the 135 statistical measures required to feed the AdaBoost method.

Figure 4.16 shows some results obtained by the alternative corrosion detector using the three versions of the AdaBoost algorithm. As can be observed, the three versions are able to successfully detect most of the corroded areas, although the outputs provided by the Modest

Algorithm 4.5 Alternative corrosion detector using AdaBoost and Law's filters responses to describe texture.

procedure CORROSION_DETECTOR_III(img, N, τ_D)
    img: input colour image
    N: patch size in pixels
    τ_D: dissimilarity threshold
    Load the 2D energy filters
    Load codewords dictionary
    Initialize *img_out* to *no corrosion*
    for all energy filter f **do**
        Compute the filter output I_f
    end for
    for all pixel p in img **do**
        for all filter output I_f **do**
            Initialize $\mathcal{V}$ as an empty array
            Get the $N \times N$ patch Π_f of I_f centred at p
            Compute the standard deviation σ_f of Π_f
            Compute the mean of positives μ_f^+ of Π_f
            Compute the mean of negatives μ_f^- of Π_f
            Save measures into $\mathcal{V}$, averaging with the measures of the rotated filter, if
              applicable
        end for
        Classify $\mathcal{V}$ using AdaBoost
        if $\mathcal{V}$ is classified as *corrosion* **then** ▷ Proceed with the colour stage
            Get the $N \times N$ patch Π of img centred at p
            Compute the 32-bin histogram for the hue channel of Π
            Compute the 32-bin histogram for the saturation channel of Π
            Stack the two histograms together into a codeword CW
            while there are more models mod in the dictionary **and**
                  p has not been labelled in *img_out* **do**
                Calculate the Bhattacharyya distance D between CW and the model mod
                if $D < \tau_D$ **then** ▷ The codeword is similar to the model
                    Label p as *corrosion* in *img_out*
                end if
            end while
        end if
    end for
    return *img_out*
end procedure

version seem to match better with the ground truth images. These results agree with the ROC curves shown in Fig. 4.15 [left].

In comparison with the main approach presented in Section 4.3.6.1, the ROC curves obtained for both approaches lie at approximately the same distance to the (0,1) corner; hence, they all present a similar performance. Regarding the PR curves, the alternative approach provides higher precision values, although this approach can not achieve 100% of recall.

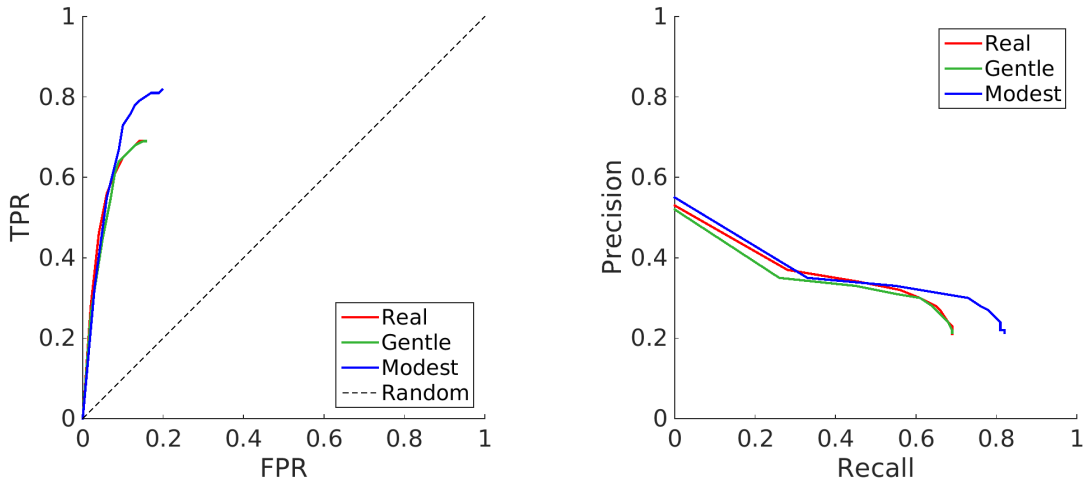


Figure 4.15: Performance of the alternative corrosion detector using Law's texture filters responses and the local stacked histograms colour model: (left) ROC curves, (right) PR curves. Performance curves obtained for the three versions of the AdaBoost method by variation of threshold τ_D.

Parameter configuration

In the following lines we summarize the parameters used in the alternative approach for corrosion detection, and discuss about its configuration. The parameters regarding the colour stage have been configured as indicated by the main approach. Regarding the parameters for the texture stage based on Law's filters responses:

- The patch size has been set to 15×15 pixels as in the case of the texture model based on the GLCM energy. This size allows a proper perception of the corrosion texture.
- The CARTs have been created comprising three levels. This low number of levels is a good compromise used to give rise to weak and fast classifiers.
- Related to the previous point, the texture stage comprises 100 weak classifiers. This value has been set high enough to improve the global performance of the texture stage (based on the combination of weak and fast classifiers) without extending the execution time unnecessarily.

4.3.7 Conclusions

Table 4.3 provides the TPR and FPR values regarding the different proposals for corrosion detection. These values have been taken from the ROC curves presented in the previous sections, selecting from the best curve the point which is closer to the (0, 1) corner. If several points are more or less situated at the same distance, we have selected the one which corresponds to a higher TPR, despite this means also a higher FPR. This is so because, for

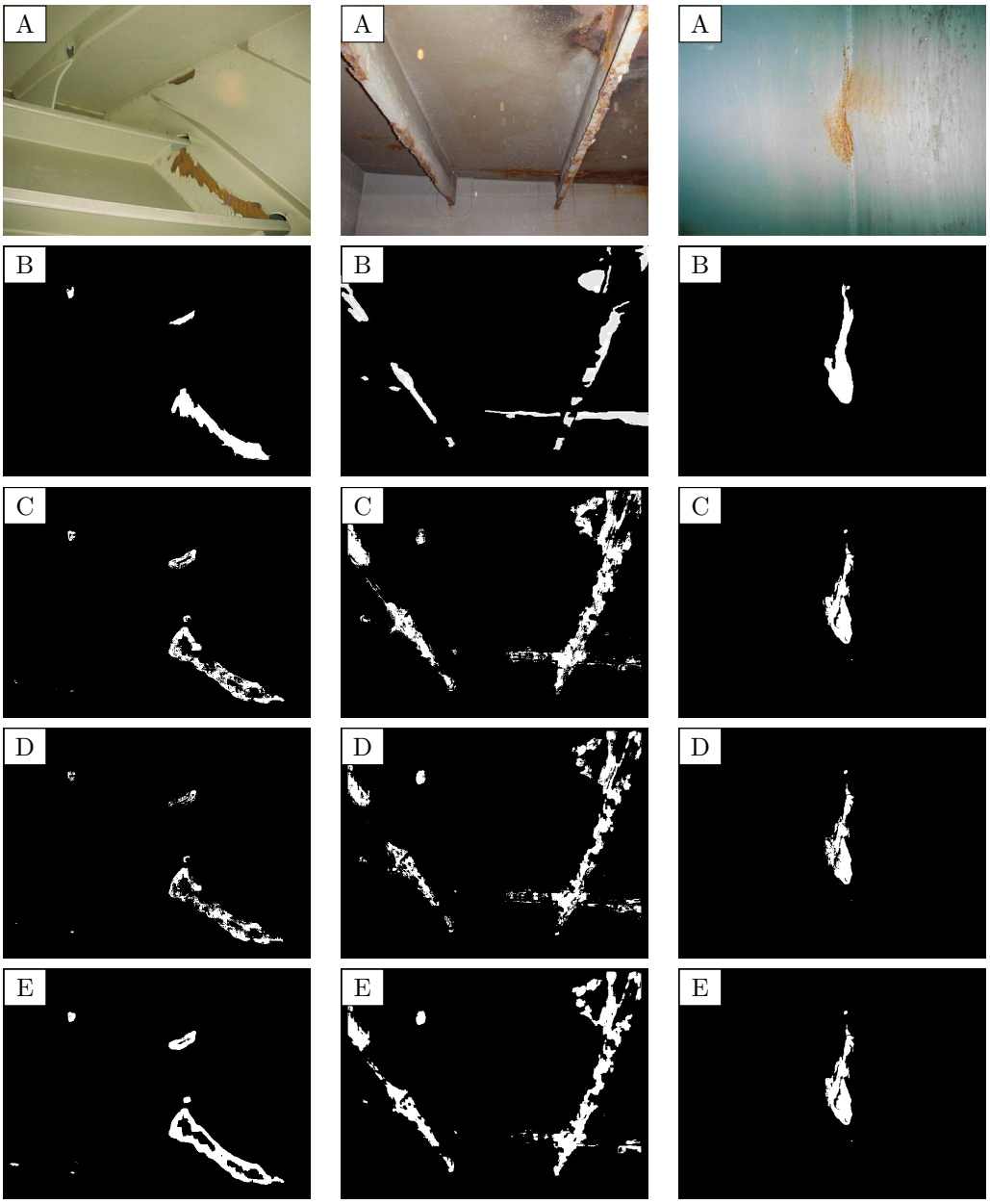


Figure 4.16: Corrosion detection results using the alternative approach: (A) input image, (B) ground truth, (C-E) corrosion detection outputs using, respectively, the Real, Gentle and Modest versions of AdaBoost for the texture stage.

our particular purpose, false positive detections are preferable to false negative detections. The table also includes the estimated execution times for processing an image with 1024×768 pixels.

Looking at these values we can conclude that the different approaches are able to detect the corroded areas in the images of the dataset. The selection of one method or another may depend on whether we prefer to obtain the best classification ratios or if we require a very

Table 4.3: Comparative assessment of the corrosion detection approaches.

	Main approach				**Alt. appr.**
	Global colour map		**Local stacked hist.**		
	RGB	**HS**	**RGB**	**HS**	
TPR	0.81	0.82	0.88	0.84	0.82
FPR	0.26	0.27	0.22	0.21	0.20
s/image*	1.57	2.33	24.47	17.50	42.69

* for a 1024×768 image.

fast answer. In the first case, the use of the local stacked histograms model using RGB for describing corrosion is recommended, in combination with the GLCM energy based texture model. In the second case, the combination of the energy model with the RGB global colour map provides the fastest classification with an acceptable accuracy.

In any case, the use of the HS colour channels reduces the memory consumption with regard to the use of the RGB channels: when using a global colour map, this comprises just 256×256 values, instead of 256×256×256; and in the case of using the codewords dictionary, each codeword comprises 32×2 values, instead of 32×3.

Regarding the use of the Law's filters responses texture model used in the alternative approach, results indicate that the texture of corroded surfaces can be learned. Nevertheless, the resulting classifier does not outperform the ones using the GLCM energy (i.e. the main approach), while it significantly increases the program complexity and the execution time.

4.4 Detection of Cracks on Vessel Structures

As reviewed in Section 2.2.1, most crack detectors are based on edge detection techniques or morphological operators. These techniques are however not directly applicable to our particular problem. As shown in Fig 4.17, cracks on vessel structures can present a large variety of lengths and appearances (see also Fig. 1.4 and Fig. 1.6). Furthermore, the distance to the inspected surface and illumination conditions in the field can be also very diverse. This makes our problem more difficult than looking for cracks in, for example, concrete surfaces under controlled conditions.

Solutions based on morphological operators, such as [133], can not be applied since they require to define a structuring element for the operator, which is tightly related to the size (width) of the crack.

Edge detection techniques can however be a good starting point for crack detection. Nevertheless, images in our dataset contain multiple elements which are part of the vessel structure that may result in edges. Furthermore, the dark shadows created when these structural elements are illuminated may be easily misclassified as cracks (see Fig. 4.17). For these reasons, it is clear that a more careful process is necessary to obtain a successful detection of cracks.

Figure 4.17: Examples of cracks on vessels structures.

Because of all the aforementioned, we propose an approach which combines edge detection with a region-growing procedure. This is performed following a set of rules which pretends to identify the different kinds of cracks which appear in the dataset images and yield a reduced number of false detections.

4.4.1 The Crack Detection Approach

This section presents a crack detector based on a region-growing procedure, similarly to the algorithm by Yamaguchi and Hashimoto described in [118]. The latter method was, however, devised for detecting cracks in concrete, what makes the authors assume a geometrical structure that does not match exactly the shape of cracks that are formed in steel. Notice that cracks on concrete are elongated and very thin, while cracks and fractures in vessel structures may present different appearances depending on their cause (see the examples of Fig. 4.17). Besides, the method presented in [118] requires specific conditions when capturing the images, including a very short and constant distance to the surface.

A region-growing procedure starts from a seed element and propagates in accordance with a set of rules. In our case, the rules are defined to identify dark, narrow and elongated sets

of connected pixels. These sets of pixels are treated as entities and are supposed to coincide with cracks.

To locate seed pixels, we first compute the edge map of the input image. The resulting edges are dilated using the corresponding morphological operator, in order to penetrate into the potential cracks. Let G_p be the mean gray level value of an $N \times N$ patch centred a pixel p, and τ_G be a threshold taking values from [0, 1]. Those edge pixels p whose gray level value is darker than $\tau_G G_p$ (i.e they are darker than its surrounding neighbourhood) are labelled as seed pixels.

For all the seed pixels found, a region is grown inside a window of $N \times N$ pixels until the window boundary is reached or no more pixels can be added. The growing proceeds through the 8-neighbouring pixels whose gray level value is below than $\tau_G G_p$, where G_p is computed for each pixel. When the window boundary is reached, the grown area is labelled as a potential crack if its elongation ϵ is larger than τ_L. The elongation is calculated by means of

$$\epsilon = \sqrt{1 - \frac{\mu_{xx} + \mu_{yy} - \sqrt{4\mu_{xy}^2 + (\mu_{xx}^2 - \mu_{yy}^2)}}{\mu_{xx} + \mu_{yy} + \sqrt{4\mu_{xy}^2 + (\mu_{xx}^2 - \mu_{yy}^2)}}}, \tag{4.14}$$

where μ_{xx}, μ_{yy} and μ_{xy} are the normalized second central moments of the region [194].

A second procedure is performed over the potential cracks found in the first step. During this process, connected potential cracks are merged into single entities. The elongation of the final elements is checked once again, which is required to be above the threshold τ_L. Furthermore, during this step, small collections of pixels are discarded as they are probably produced by noise in the image.

The flowchart of the complete algorithm can be found in Figure 4.18 while the pseudocode is available as Alg. 4.6. This flowchart does not comprise a pre-processing step that has been incorporated to reduce the noise of the image. This noise filtering step is implemented using a Bilateral filter [195] due to its excellent properties for preserving relevant edges.

As indicated in the pseudocode, and in order to save time, the mean gray-scale value for the $N \times N$ neighbourhood of each pixel (G_p) is pre-computed in a previous step and saved to *img_mean*. This is performed by convolving the image with an $N \times N$ box kernel.

The values for the different parameters have been set to maximize crack detection and keep the false detections contained. The window size N is set as a percentage of the minimum of the image height and the image width. This is so to tolerate different image resolutions. Notice that, despite this parameter is used to indicate the range for the region-growing process, this is intended to detect portions of cracks, so that the algorithm is able to detect larger cracks by combining different crack portions. In other words, the length of cracks is not limited by the window size N.

Regarding the edge detection process, the Canny operator [124] has been used. This is a well known method which has been proved useful to our purposes has also been used by other authors to implement their crack detection algorithms (see Section 2.2.1 for some examples).

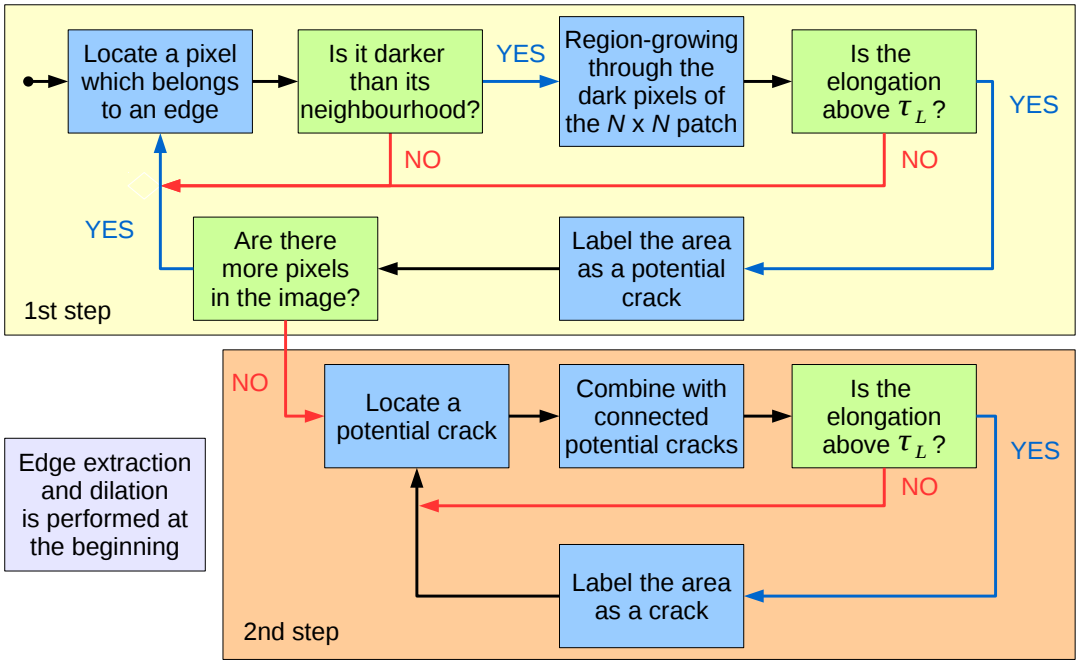


Figure 4.18: Flowchart of the crack detector.

Finally, the thresholds τ_G and τ_L, which are used to specify the gray-level intensity and the elongation of the crack, have been set to 0.8 and 0.35 respectively, since these values provide good results with the majority of the images of the dataset.

4.4.2 Corrosion-guided Crack Detector

In order to improve the performance of the crack detector, a modified version is proposed, which combines corrosion detection with the region-growing strategy. The rationale behind this approach lies on the observation that most of the cracks in metallic surfaces appear in corroded areas. Therefore, a successful corrosion detection can be used to guide the localization of cracks in the images.

To guide the crack inspection, we have selected the main corrosion detection approach presented in Section 4.3.6.1. This has been configured to combine RGB local stacked histograms with the GLCM energy, since this configuration has resulted into the best detection ratios (see Section 4.3.7).

To implement the guided crack detection, the initial condition to start a region-growing has been slightly modified so that it considers a seed pixel only if it has been labelled as corroded. The original conditions, which check whether the pixel is over an edge and if it is darker than its neighbourhood, are also required.

For this crack detection approach, the corrosion detector has been configured to provide a very low FPR, so that all the pixels provided to the crack detector are likely corroded. To

Algorithm 4.6 Crack detector.

procedure CRACK_DETECTOR(img, γ, τ_G, τ_L)
    img: Bilateral-filtered gray-scale image
    γ: parameter used to define the window size N
    τ_G, τ_L: gray-level and elongation thresholds
    Initialize *img_out* to *no crack*
    Define N as $\gamma \cdot \min(height, width)$
    Compute the filtered image *img_mean* using an $N \times N$ window
    Compute the *edge_map* of *img*
    Dilate the edges of the *edge_map*
    for all pixel p in *img* **do**
        $F = \emptyset$ ▷ Currently flooded area
        $G_p = img\_mean(p)$ ▷ Mean value of the $N \times N$ patch centred at p
        if p is darker than τ_G G_p **and** p is an edge **then** ▷ p is a seed pixel
            $F = F \cup \{p\}$
            while F does not reach $N \times N$ boundary **do** ▷ Region-growing process
                for all neighbours q of F **do**
                    $G_q = img\_mean(q)$ ▷ Mean value of the $N \times N$ patch centred at q
                    if q is darker than τ_G G_q **then**
                        $F = F \cup \{q\}$
                    end if
                end for
            end while
            if *Elongation(F)* $> \tau_L$ **then**
                Label F as a *crack* in *img_out* ▷ Potential crack
            end if
        end if
    end for
    for all potential cracks C in *img_out* **do**
        Merge with other connected potential cracks
        if *Elongation(C)* $< \tau_L$ **or** C is very small **then**
            Label C as *no crack* in *img_out* ▷ Discard potential crack
        end if
    end for
    return *img_out*
end procedure

attain this, the thresholds τ_E and τ_D are set to 0.4 and 0.1 respectively, obtaining a FPR of 0.07 and a TPR of 0.55. The moderate TPR is not an issue since only a small collection of pixels is required to be used as seeds.

4.4.3 Evaluation of the Crack Detectors

As mentioned before, to evaluate the performance of the crack detectors, we have considered cracks as entities, so that we have assessed whether the algorithm is able to detect at least part of all the cracks in the dataset (see Section 4.2 for details). During these assessment,

Table 4.4: Precision and recall values of the original and guided versions of the crack detector.

	Original	**Guided**
Precision	0.60	0.75
Recall	1	0.84

Table 4.5: Precision and recall values of the original and guided versions of the crack detector after removing four images from the dataset.

	Original	**Guided**
Precision	0.74	0.87
Recall	1	1

the capability to provide a low number of false positives has been also valued. In this regard, Table 4.4 provides the precision and recall values obtained using both the original and the corrosion-guided versions of the crack detector.

As can be seen, the original version of the algorithm is able to detect all the cracks in the dataset (recall is 1), while 40% of the cracks indicated by the algorithm are false positives (precision is 0.6). When using the corrosion-guided version, precision increases to 0.75, so that the false positive detections are reduced to 25%. Nevertheless, the guided version is not able to detect all the cracks in the dataset. Nevertheless, this behaviour takes place because the dataset comprises four images with cracks which do not present any corroded point. Table 4.5 presents the performance metrics obtained for the dataset when these four images are not considered. In this case, the guided version is able to reach 100% of recall. Regarding the new precision values, both the original and corrosion-guided versions obtain a higher score, attaining 0.75 and 0.87 respectively. This indicates that the four images which have not been considered produce several false positive detections.

Regarding the execution times, Table 4.6 provides the μs per pixel corresponding to both versions of the algorithm. The table also provides the estimated seconds to process a 1024×768 image. Notice that, as happened in the case of corrosion, this value is just an estimation computed averaging the different observations, and the actual execution time will depend on the amount of defective pixels in the image. As indicated in the table, the original crack detector would take 0.63 s to analyse an image with 1024×768 pixels, while the guided version would need more than 26 s. This difference is clearly due to the execution of the corrosion detector.

Figure 4.19 shows some results obtained using both versions of the crack detector. As can be observed, the two versions are able to detect the cracks in the left and middle images, where the guided version provides less false positives. The image on the right is one of the four images where the guided version is not able to detect the crack, since it is not affected by corrosion.

A last experiment has been performed in order to provide further results regarding the

Table 4.6: Execution times of the original and guided versions of the crack detector.

	Original	**Guided**
μs/pixel	0.81	33.78
s/image*	0.63	26.43

* considering a 1024×768 image.

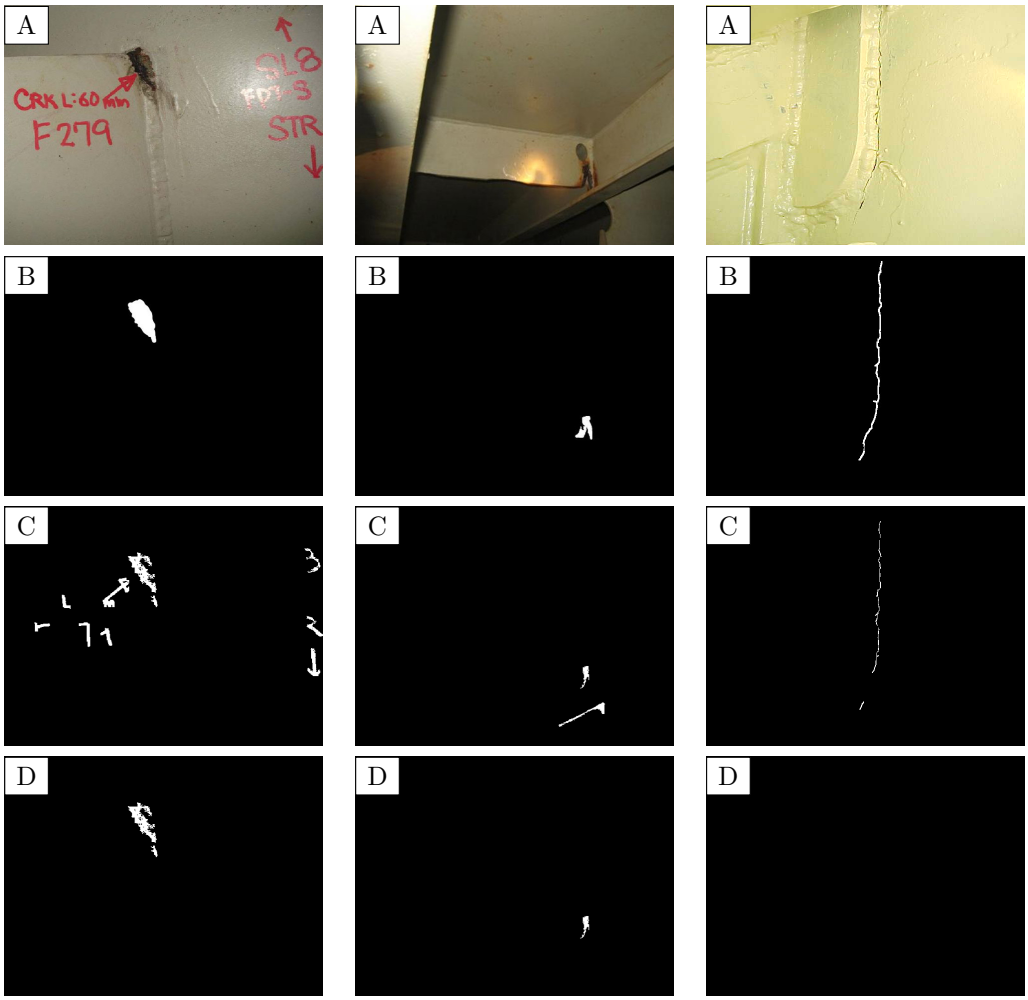


Figure 4.19: Crack detection results for some images of the dataset: (A) input image, (B) ground truth, (C-D) crack detection using, respectively, the original and the guided versions of the method.

crack detector performance. In this respect, an additional dataset including cracks on fiberglass boats have been evaluated. This dataset, comprising 24 pictures, has not been used during the design of the crack detection algorithm since cracks on fiberglass do not look the same as cracks on metallic structures. Nevertheless, this is used here to show the good performance of our method in a different scenario. The results provided in Fig. 4.20 show

that cracks are mostly detected, despite they present diverse shapes and are captured from different distances and with different illumination conditions.

Parameter configuration

By way of summary, the parameters of the crack detector have been configured taking into account the following considerations:

- The gray-scale threshold τ_G has been configured to 0.8. This value forces crack-affected pixels to be 20% darker than their surrounding pixels. If this threshold is set too low, those cracks situated in dark areas of the image are not successfully detected. On the contrary, a too high value can lead to non-defective pixels to be considered as belonging to a crack.

- The elongation threshold τ_L has been set to 0.35. This value allows the detection of the majority of the cracks in the image dataset. Notice that a too low value would result in the detection of just the very thin and elongated cracks (see for example Fig. 4.17 [bottom-left]), while short fractures, such as the one shown in Fig. 4.17 [top-left], would not be detected. On the contrary, a too high value would result in the miss-classification of non-defective collections of dark pixels.

- When using the corrosion detector to guide the inspection, this is configured as indicated in the corresponding section. The discrimination thresholds τ_E and τ_D have been set to 0.4 and 0.1 respectively. These values ensure that a moderate collection of actually corroded pixels is available to be used as seeds for the region-growing procedure.

4.4.4 Conclusions

Crack detection on vessel structures is a challenging task due to the diversity of appearances that cracks can present, and the different locations where they can occur. We have presented a crack detection approach that deals with this problem. Contrary to existing methods, our approach does not require a fixed distance to the inspected surface, neither supposes any specific crack size. Just the opposite, our method is devised to detect dark and elongated collections of pixels, which may present different widths, and where "dark" means "darker than the surrounding area". The result is an algorithm which is able to detect all the cracks in the dataset.

Optionally, a modification of the algorithm can be activated which allows to reduce the false positive detection. Nevertheless, this improvement, based on a prior search for corroded areas, may fail with cracks which are not affected by corrosion. This situation, which occurs just in 17% of the images in the dataset, should be taken into account to decide whether using the guided version or not. The coating conditions and the general state of the metallic

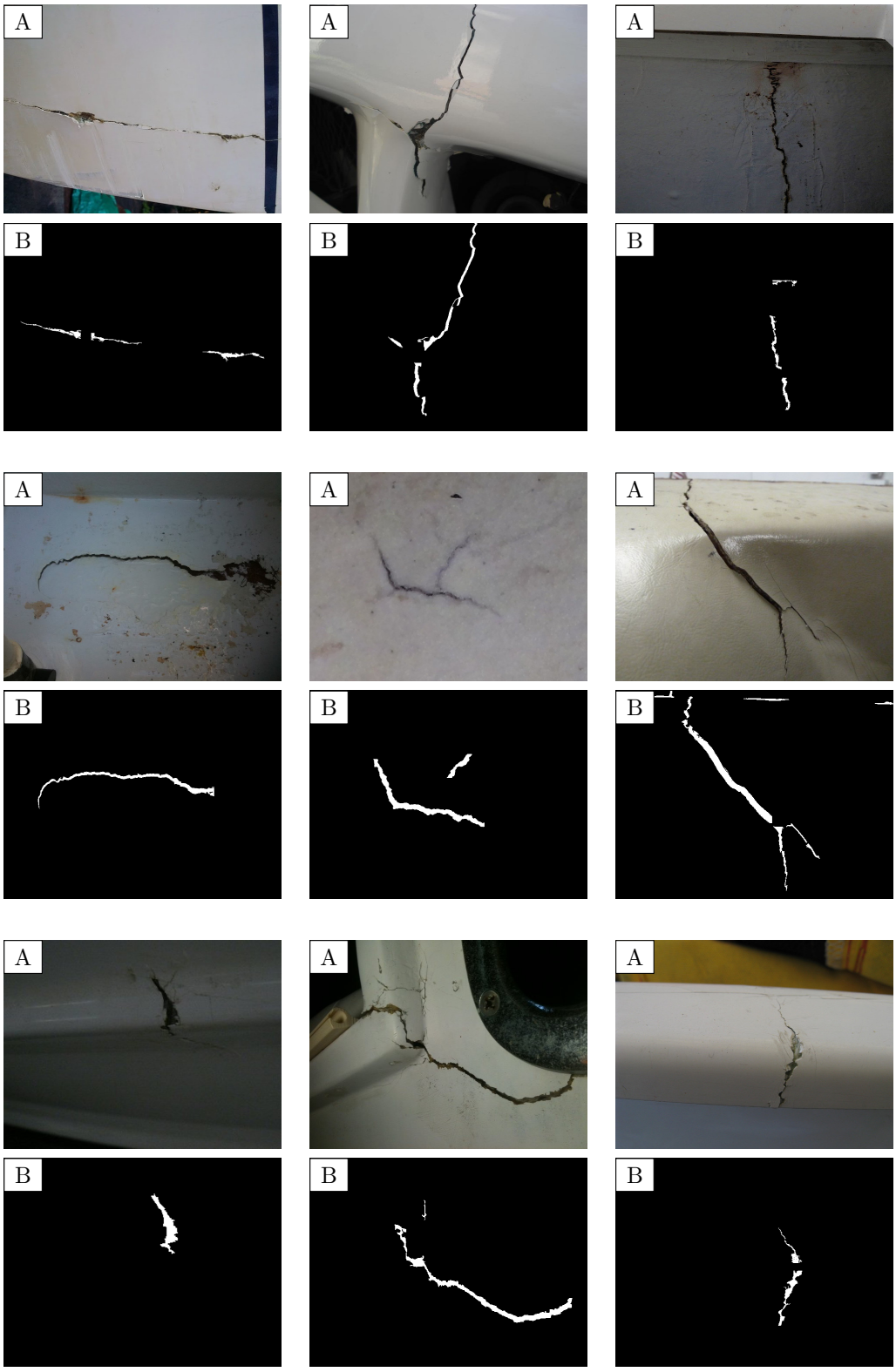


Figure 4.20: Detection of cracks on fiberglass boats: (A) input image, (B) output of the crack detector.

plates under inspection can be a possible indicator to recommend whether to use the original version, or the corrosion-guided version.

Finally, the good results detecting cracks on fiberglass boats indicate that our algorithm is flexible enough to detect cracks in pictures taken from other non-metallic materials.

4.5 Saliency-inspired Algorithms for General Defect Detection on Vessel Structures

4.5.1 Overview

Specific defect detectors to identify one of the two main defects on vessel structures, i.e. cracks and corrosion, have been described in the previous sections. As an alternative way to approach the detection problem, this section deals with the idea of detecting generic, instead of specific, defects.

Vessel structures consist of large surfaces that usually present a regular texture. When these surfaces are inspected from a certain distance, a defect appears as a discontinuity that alters the regularity of the texture. Based on that, texture-related features seem to be a good option to differentiate between defective and non-defective areas. Furthermore, defects can also be considered as rare phenomena that may appear on such regular surfaces. Since they are rare, defects become potential attractors of the visual attention of the surveyor during a visual inspection process. Following these ideas, we propose to use texture-based features typically used in cognitive models to predict human eye fixations, also known as *saliency models*.

Within these saliency models, image saliency operators are used to characterize some parts of a scene that, for an observer, would stand out relative to their neighbouring areas. Although the term "salient" is often considered in the context of scene-driven bottom-up computations, in this work we also consider the combination of bottom-up information with the results of expectation-driven top-down computations.

The related literature contains several saliency models based on different principles and devised for a variety of purposes [196]. Among them, we focus on those which, once evaluated, result into a saliency map. A saliency map consists in a topographic representation of the conspicuousness of the different areas of the input image [197]. This can be shown as a gray-scale image where locations with higher conspicuity values are closer to white and less salient areas are closer to black.

In the following sections we provide our answer to the basic questions to be answered when designing a classifier:

1. which features are the best for a suitable classification,

2. how many features are necessary, and

3. how should these be combined to implement the best classifier.

Firstly, Section 4.5.2 presents the saliency-related features selected for detecting defects on vessel structures. After that, in Section 4.5.3 and Section 4.5.4 we propose two frameworks to combine these features to perform the defect detection. On the one hand, the first approach consists in a flexible and generic framework which allows to easily combine data provided by different features. The way how this combination is performed can be specified, what allows exploring multiple combination operators. On the other hand, a Bayesian framework is used in the second approach to combine feature data. This framework allows introducing not only bottom-up information, but also top-down data. Both frameworks are evaluated and their performances are compared in Section 4.5.5. Finally, some conclusions are stated in Section 4.5.6.

4.5.2 Contrast and Symmetry as Salient Features for Defect Detection

As indicated in [196], three features have been traditionally used in computational models of attention: intensity, colour and orientation. The sudden variation of some of these features, computed as a local contrast, increases the conspicuousness of the area producing bottom-up guidance [198].

The information resulting from the variation of these three features is typically combined into a single contrast-based saliency map (see for example [199–202]). The first approach describing a saliency model which combines contrast in intensity, colour and orientation was presented by Itti et al. [203]. This model, which is inspired in the behaviour and neuronal architecture of the early primate visual system, has become one of the most influential saliency computation approaches following a model-based paradigm, and has inspired later authors who have contributed with their own saliency models (see [196]). Following this line, we propose to use the local contrast (combining intensity, colour and orientation) as a first feature to locate defects on vessel structures.

A saliency model based on the Gestalt principle of symmetry was presented in [204]. In this work, the authors discuss local symmetry as a measure of saliency and investigate its role in visual attention. To this end, they use three different symmetry operators (isotropic, radial and colour symmetry operators) and compare them with human eye tracking data. Their results suggest that symmetry is a salient structural feature for humans, as well as the suitability of their method for predicting human eye fixations in complex photographic images, where symmetry is not so evident.

Furthermore, the authors use the saliency model by Itti et al. as a reference for comparison. Their results show that, on many occasions, their symmetry operators outperformed the contrast-saliency model. These are the reasons why we decided to incorporate symmetry as a second feature for defect detection on vessels.

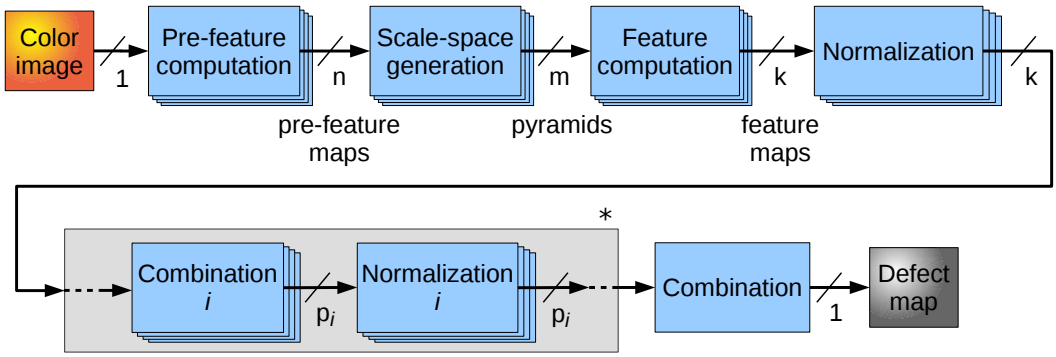


Figure 4.21: Generic framework for defect detection. * refers to zero or more than zero instances of the corresponding stage.

4.5.3 A Generic Framework for Defect Detection

We oriented the design of our first generic defect detector towards a flexible framework which allows an easy integration of different features and their combinations. To attain this level of flexibility, we consider that the framework must cover the following aspects:

1. it should allow computing one or more features that are potentially useful to discriminate between defective and non-defective situations,

2. final features response should not depend on scale,

3. one or more combination operators should be available to merge the information provided by the computed features and try to find the combination (if any) that improves the classification performance, and

4. related to the previous point, one or more normalization operators should be available to adapt the different features responses to a certain range, in order to ensure a proper combination.

This generic framework is organized as a modular pipeline which involves different stages that can be configured (or even removed) depending on our needs, so that different configurations result into different defect detectors. The pipeline is provided in Fig. 4.21. Within the framework, each feature is intended to be computed as a different thread and the information they all provide to be combined to make up the final detection output.

In more detail, the framework consists of the following stages:

- *Pre-feature computation.* The first stage prepares the input image to provide the information necessary to compute all features. From an input colour image, one can obtain, for example, the gray-scale (or intensity) image, the red channel image, the saturation

image (from HSV colour space), etc. Each one of these images is called a *pre-feature map*.

- *Scale-space generation.* This stage scales the pre-feature maps using a range of scale factors to obtain a collection of multiple-scale representations, also known as pyramids. The computation of each pyramid level can include filtering the input map using a specific kind of filter. One can compute, for example, a Gaussian pyramid which progressively low-pass filters and sub-samples the pre-feature map, an oriented Gabor pyramid for a preferred orientation α, a simple sub-sampling pyramid computed without any filtering, etc.

- *Feature computation.* This is the core stage within the pipeline. Each instance of this stage is in charge of computing the value for a given feature for all the pixels of the input image. Since this can be fed with one or more multi-scale pyramids, a feature can be computed combining the information provided at different scales. Every output of this stage is called a *feature map*.

- *Normalization.* This step normalizes the different feature maps to the same range of values to enable their combination.

- *Combination.* This is the last stage of the pipeline. It is in charge of combining the normalized feature maps in order to obtain a single map, which is called the *defect map*. The mean and the median operators are some examples of simple combination operators. Unary operators such as unary minus or thresholding are also be considered.

As indicated in Fig. 4.21, the generic framework allows computing more complex features by means of concatenating multiple instances of normalization and combination stages.

The output of the framework is the defect map, which consists in a single-channel map where defective areas are supposed to be labelled with higher values. Notice that this representation fits with the definition of saliency map.

Using this generic framework, several defect detectors have been implemented employing the features indicated in Section 4.5.2. Furthermore, in order to merge the information provided by the single features to improve the detection performance, different combination operators have been evaluated. The resulting detectors are described in the following.

Contrast-based Detector

The generic framework stated in Fig. 4.21 has been firstly used to design a contrast-based defect detector. The model presented in [203] has been used as source of inspiration. The pipeline is detailed in Fig. 4.22. As for its implementation, each one of the stages of the generic pipeline of Fig. 4.21 is particularized as follows:

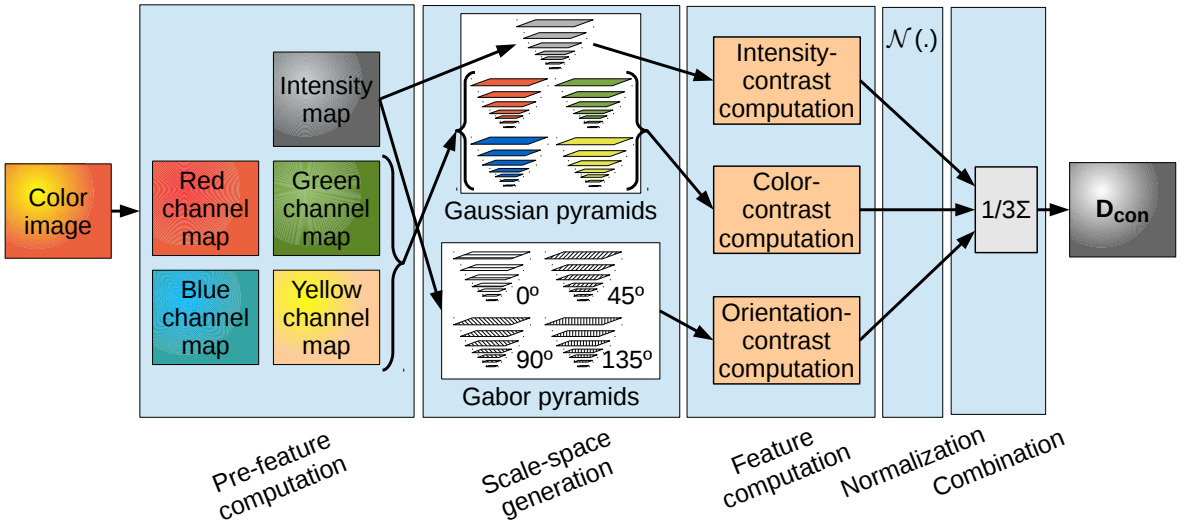


Figure 4.22: Implementation of the contrast-based defect detector using the generic framework.

- *Pre-feature computation.* Five pre-feature maps are computed from the red (r), green (g) and blue (b) colour channels of the input image:

$$I = \frac{r+g+b}{3}, \tag{4.15}$$

$$R = r - \frac{g+b}{2}, \tag{4.16}$$

$$G = g - \frac{r+b}{2}, \tag{4.17}$$

$$B = b - \frac{r-g}{2}, \tag{4.18}$$

$$Y = \frac{r+g}{2} - \frac{|r-g|}{2} - b, \tag{4.19}$$

  where I is the intensity map, R is the red channel map, G is the green channel map, B is the blue channel map and Y is the yellow channel map. During the computation of these maps, negative values (if any) are set to zero.

- *Scale-space generation.* Nine pyramids are computed from the pre-feature maps. On the one hand, five Gaussian pyramids $\widehat{I}$, $\widehat{R}$, $\widehat{G}$, $\widehat{B}$ and $\widehat{Y}$ are computed by progressively low-pass filtering and sub-sampling the pre-feature maps (I, R, G, B and Y). On the other hand, four Gabor pyramids $\widehat{O}_0$, $\widehat{O}_{45}$, $\widehat{O}_{90}$ and $\widehat{O}_{135}$ are computed filtering the intensity

pyramid $\widehat{I}$ with oriented Gabor filters with orientations $\alpha \in \{0°, 45°, 90°, 135°\}$. All pyramids comprise seven scales, ranging from 1:1 (scale one) to 1:64 (scale seven).

- *Feature computation.* Three threads are executed in parallel to build three feature maps, respectively corresponding to the contrast level in intensity ($\boldsymbol{I}$), colour ($\boldsymbol{C}$) and orientation ($\boldsymbol{O}$). This computation is performed as indicated in [203]. A first step computes center-surround differences between fine and coarse scales from the pyramids; that is to say, it computes the difference between each pixel of a fine (or center) scale c and its corresponding pixel in a coarse (or surrounding) scale s. Accordingly, preliminary maps $\boldsymbol{I}(c,s)$, $\boldsymbol{RG}(c,s)$, $\boldsymbol{BY}(c,s)$ and $\boldsymbol{O}(c,s,\alpha)$ result as follows:

$$\boldsymbol{I}(c,s) = |I(c) \ominus I(s)|, \tag{4.20}$$

$$\boldsymbol{RG}(c,s) = |R(c) - G(c)) \ominus (G(s) - R(s))|, \tag{4.21}$$

$$\boldsymbol{BY}(c,s) = |(B(c) - Y(c)) \ominus (Y(s) - B(s))|, \tag{4.22}$$

$$\boldsymbol{O}(c,s,\alpha) = |O(c,\alpha) \ominus O(s,\alpha)|, \tag{4.23}$$

where $|x|$ refers to the absolute value of x, $\ominus$ is the across-scale subtraction operator (see Fig. 4.23), $\boldsymbol{I}(c,s)$ accounts for the intensity contrast, $\boldsymbol{RG}(c,s)$ accounts for red/green contrast, $\boldsymbol{BY}(c,s)$ accounts for blue/yellow contrast and $\boldsymbol{O}(c,s,\alpha)$ accounts for the orientation contrast for a given orientation α. In our implementation, the scales are defined as $c \in \{1,2,3\}$ and $s = c + \delta$, with $\delta \in \{3,4\}$.

In a second step, the intermediate maps are combined into the following three feature maps by means of the across-scale addition operator $\oplus$ (see Fig. 4.23 for details):

$$\boldsymbol{I} = \bigoplus_{c=1}^{3} \bigoplus_{s=c+3}^{c+4} \mathcal{N}(\boldsymbol{I}(c,s)), \tag{4.24}$$

$$\boldsymbol{C} = \bigoplus_{c=1}^{3} \bigoplus_{s=c+3}^{c+4} \left(\mathcal{N}(\boldsymbol{RG}(c,s)) + \mathcal{N}(\boldsymbol{BY}(c,s))\right), \tag{4.25}$$

$$\boldsymbol{O} = \sum_{\alpha \in \{0°,45°,90°,135°\}} \mathcal{N}\left(\bigoplus_{c=1}^{3} \bigoplus_{s=c+3}^{c+4} \mathcal{N}(\boldsymbol{O}(c,s,\alpha))\right), \tag{4.26}$$

where $\mathcal{N}(.)$ is a normalization operator devised to promote high and isolated peaks. It adjusts the map to a fixed range [0, M] and multiplies it by $(M - \overline{m})^2$, being $\overline{m}$ the average of all local maxima that do not coincide with the global maximum.

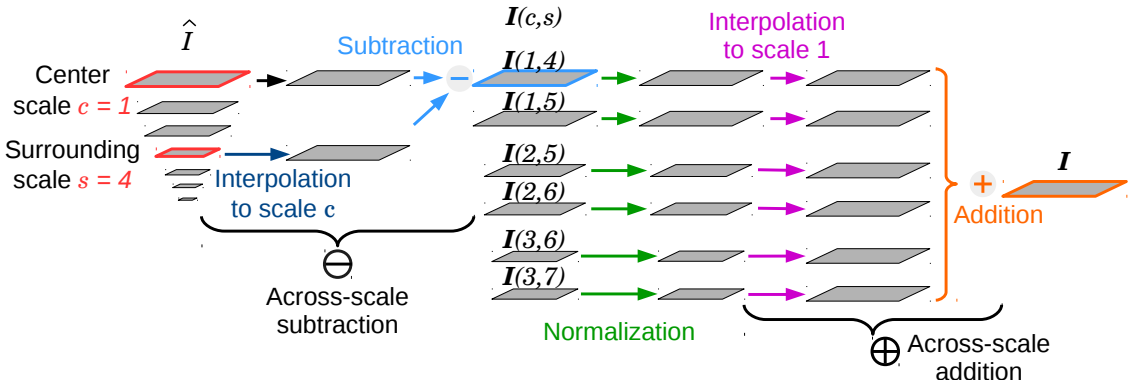


Figure 4.23: Illustration of feature map computation: case of intensity-contrast map.

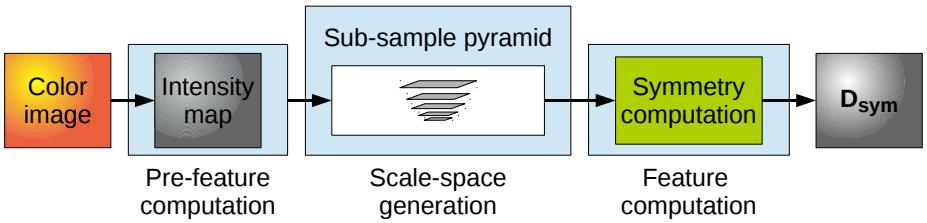


Figure 4.24: Implementation of the symmetry-based defect detector using the generic framework.

By way of illustration, a diagram showing the entire feature computation for map $\boldsymbol{I}$ can be found in Fig. 4.23.

- *Normalization.* The normalization operator $\mathcal{N}(.)$ is used now to promote the highest and isolated peaks in the three feature maps, obtaining $\overline{\boldsymbol{I}}$ for intensity, $\overline{\boldsymbol{C}}$ for colour and $\overline{\boldsymbol{O}}$ for orientation.

- *Combination.* The final defect map is computed using a linear combination:

$$\boldsymbol{D}_{con} = \frac{\overline{\boldsymbol{I}} + \overline{\boldsymbol{C}} + \overline{\boldsymbol{O}}}{3}, \tag{4.27}$$

so that any salient point in any of the feature maps appears in the final defect map.

Symmetry-based Detector

The generic framework (Fig. 4.21) has been also configured to implement a symmetry-based defect detector. This is provided as Fig. 4.24. Each stage of the generic framework is particularized as follows:

- *Pre-feature computation.* It computes one intensity map as indicated in Eq. 4.15.

- *Scale-space generation.* This stage computes a simple sub-sampling pyramid with five scales, ranging from 1:1 (scale one) to 1:16 (scale five).

- *Feature computation.* The symmetry map is calculated for each level l of the pyramid, using the isotropic operator [205]. We have chosen this operator because it is easier to compute and no significant improvement was observed when using the radial or colour symmetry operators proposed in [206] for predicting human eye fixations.

  To obtain the final defect map based on symmetry, the five responses $\boldsymbol{M}(l)$ (one per pyramid level) are normalized using the normalization operator $\mathcal{N}(.)$ and finally added together across-scale into a scale 1:1 map:

$$\boldsymbol{D}_{sym} = \bigoplus_{l=1}^{5} \mathcal{N}(\boldsymbol{M}(l)). \tag{4.28}$$

Normalization and combination stages are not employed for this case since symmetry is the only feature used.

Contrast and Symmetry Combination

In order to deeper explore the possibilities of the selected features, the generic framework has also been configured to combine the information that they convey in the following way:

- *Pre-feature computation.* Five pre-feature maps are computed as described for the contrast-based method.

- *Scale-space generation.* It generates ten pyramids, nine used for contrast and one used for symmetry, as detailed in previous sections.

- *Feature computation.* It consists of four threads, one for each channel of contrast (intensity, colour and orientation) and one for symmetry. They proceed as indicated in previous sections.

- *Normalization.* The normalization operator $\mathcal{N}(.)$ is used in this stage to promote the areas of the feature maps that have been indicated as potentially defective by any of the features. Therefore, $\overline{\boldsymbol{D}_{con}}$ is obtained as the normalized version of the defect map based on contrast and $\overline{\boldsymbol{D}_{sym}}$ is the analogue for the case of symmetry.

- *Combination.* We initially propose two operators. The first one consists in a linear combination of the contrast and symmetry-based defect maps:

$$\boldsymbol{D}_{OR} = \frac{\overline{\boldsymbol{D}_{con}} + \overline{\boldsymbol{D}_{sym}}}{2}. \tag{4.29}$$

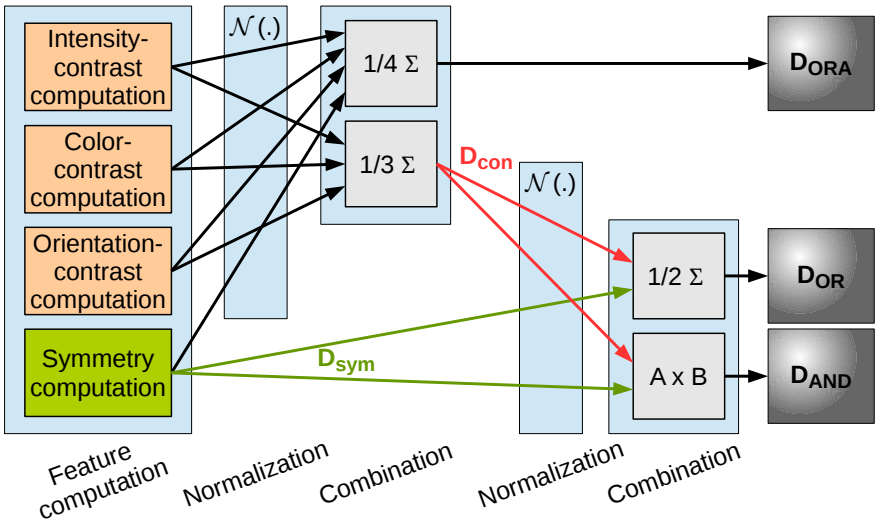


Figure 4.25: Set up of the normalization and combination stages for the defect detectors merging contrast and symmetry information.

This combination allows any defective point in any of the maps to be promoted so that it stands out in the final defective map. From now on, it will be referred to as the *OR* combination.

The second combination operator that we propose merges the contrast and symmetry-based defect maps so that defective regions in the resulting map are required to be simultaneously indicated as potentially defective in both maps, implementing, in a certain sense, the logical *AND* operator:

$$\boldsymbol{D}_{AND} = \overline{\boldsymbol{D}_{con}} \times \overline{\boldsymbol{D}_{sym}}. \tag{4.30}$$

In addition to these combinations, a third version has been considered which intends to explore the contribution provided by the different contrast channels, i.e., intensity, colour and orientation. The four feature maps (including the symmetry map) are fused using a modified version of the OR combination, which will be referred to as the *ORA* (Or-Alternative) combination:

$$\boldsymbol{D}_{ORA} = \frac{\overline{\boldsymbol{I}} + \overline{\boldsymbol{C}} + \overline{\boldsymbol{O}} + \overline{\boldsymbol{D}_{sym}}}{4}. \tag{4.31}$$

Figure 4.25 shows the set up of the normalization and combination stages for the three detectors which combine contrast and symmetry information.

4.5.4 Defect Detection using a Bayesian Framework

As mentioned in Section 4.5.1, we consider defects on vessels as rare phenomena that may appear on a regular surfaces or structures. Since they are rare, the probability of an area of being affected by some kind of defect is rather low. Following this idea, we propose a second detection framework which makes use of this low probability to indicate salient areas in pictures taken from vessel structures.

A similar idea, but applied to the detection of general targets, is used by Zhang et al. [207]. These authors propose a Bayesian framework that incorporates top-down information with bottom-up saliency (self-information of visual features) to provide the pointwise mutual information between the features and the target, when searching for a target. They call their framework *Saliency Using Natural Statistics* (SUN) since they focus on learned statistics from natural scenes.

In our approach, we compute the probability of contrast and/or symmetry features and introduce them using the SUN framework. In the original framework, saliency at a given point z is defined as:

$$S_z = \underbrace{\frac{1}{p(F=f_z)}}_{\substack{\text{Independent}\\ \text{of target}\\ \text{(bottom-up saliency)}}} \underbrace{\underbrace{p(F=f_z|C=1)}_{\text{Likelihood}}\underbrace{p(C=1|L=l_z)}_{\text{Location prior}}}_{\substack{\text{Dependent on target}\\ \text{(top-down knowledge)}}} \tag{4.32}$$

where F refers to the visual feature(s) at a point, f_z represents the feature values observed at z, L is the location (pixel coordinates) of a point, l_z represents image location of z, and C denotes the class (1 = target class).

In our case, since defects do not depend on their location in the image, the formulation can be simplified to:

$$S_z = \frac{1}{p(F=f_z)} p(F=f_z|C=1) \tag{4.33}$$

Using this formulation, the saliency of a given point z decreases as the probability of features f_z is higher, and increases as the probability of f_z in defects increases. To estimate these probabilities, the Parzen windows method [188] has been applied to the histograms obtained for the different features computed for all the images of a training set (further explained in Section 4.5.5).

Similarly as for the first framework, different defect detectors have been implemented depending on the feature/s used to feed the Bayesian framework. On the one hand, we have implemented the contrast and symmetry single-feature detectors. To do that, the framework has been fed with the probabilistic data obtained using the corresponding feature. On the other hand, the combined detector has also been implemented using the probabilistic information obtained for both features. In all cases, the contrast and symmetry values for each

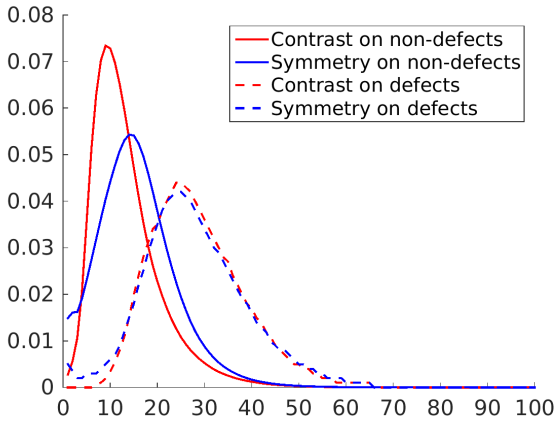


Figure 4.26: Estimated PDFs for contrast and symmetry features.

pixel of the image have been computed using the pipelines detailed in Section 4.5.3 (Fig. 4.22 and Fig. 4.24).

Furthermore, the Bayesian framework has made possible to evaluate the contribution of combining top-down knowledge with bottom-up saliency. To do that, the detection performance obtained for the detector specified through Eq. 4.33 has been compared with a simplified version which just considers bottom-up saliency:

$$S_z = \frac{1}{p(F = f_z)}. \tag{4.34}$$

4.5.5 Experimental Assessment

Prior to evaluating the performance of the defect detectors, we have assessed how suitable are contrast and symmetry to discriminate between defective and non-defective areas. To this end, the probability distribution of these two features has been computed for the two classes, i.e. defective and non-defective area, for all the images in the dataset. To estimate underlying probability density functions (PDF), we have applied the Parzen windows method [188] to the histograms corresponding to the combinations contrast/defect, symmetry/defect, contrast/non-defect and symmetry/non-defect. Figure 4.26 provides the resulting PDFs, where the contrast and symmetry values have been normalized between 0 and 100 to facilitate the comparison. From these plots, we can state the following:

- non-defective pixels present low values of contrast and symmetry (below 10 for contrast and around 15 for symmetry), what confirms that non-defective areas in vessel structures exhibit a rather uniform-intensity texture;

- defective pixels tend to present higher values of both features (around 25), so that these features seem to be useful to discriminate between defective and non-defective areas;

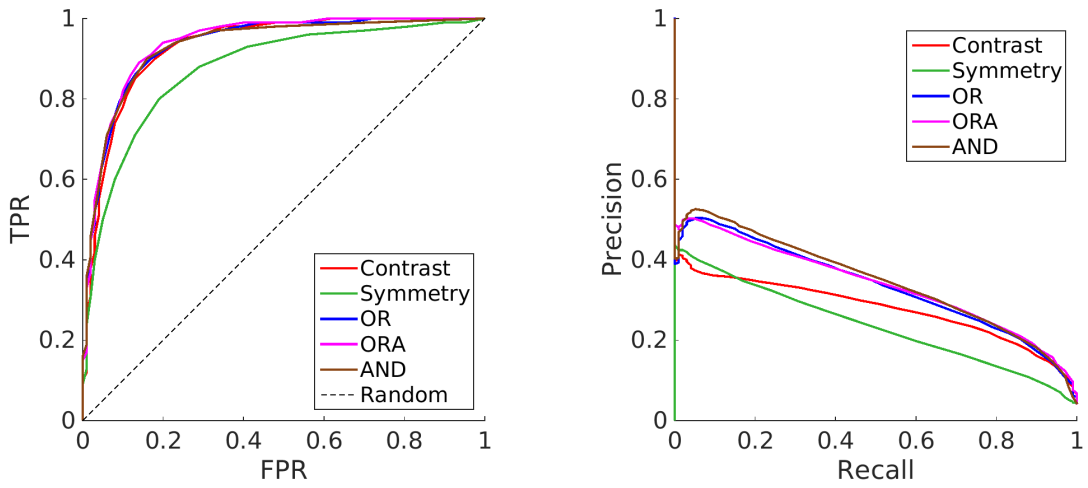


Figure 4.27: Performance of the defect detector using the generic framework: (left) ROC curves and (right) PR curves. Performance curves obtained for the five configurations of the framework, by variation of the saliency threshold τ_S.

- contrast peaks are farther from one another than symmetry peaks, what would indicate that contrast is more discriminative than symmetry when trying to distinguish the defective areas that appear in our dataset from the non-defective areas.

Similar PDFs have been computed during the training step of the SUN-based framework. These have been computed following the LOOCV methodology, as explained in Section 4.2. Notice that the Bayesian framework additionally requires computing the probability density functions for all the pixels (including both defective and non-defective) of the images in the training set, while it does not requires the PDFs for the non-defective pixels (see Eq. 4.33).

Figure 4.27 provides the ROC and PR curves for the different defect detectors implemented using the generic framework, i.e. the bottom-up approach. These curves have been generated by variation of the saliency threshold τ_S, which is used to binarize the resulting saliency map.

As can be observed, the different detectors provide good detection performances according to their ROC curves. The contrast-based detector performs better than the symmetry-based detector, what confirms that contrast is more discriminative when describing the defective areas that appear in the dataset.

Regarding the three detectors which combine both features, all three provide slightly better results than the contrast-based detector, according to their ROC curves. The improvement can be better evaluated looking at the AUC values reported in Table 4.7. Furthermore, the PR curves show that the three combined detectors attain higher precision values than the single-feature detectors (see Fig. 4.27 [right]).

Similarly, the performance of defect detectors using the Bayesian framework has also been evaluated. In this regard, Fig. 4.28 compares the results obtained when the framework uses only bottom-up saliency, with the results obtained when top-down knowledge is also

Table 4.7: AUC values for the defect detector using the generic framework. These values correspond to the ROC curves of Fig. 4.27.

	Contrast	**Symmetry**	**OR**	**ORA**	**AND**
AUC	0.930	0.876	0.935	0.940	0.932

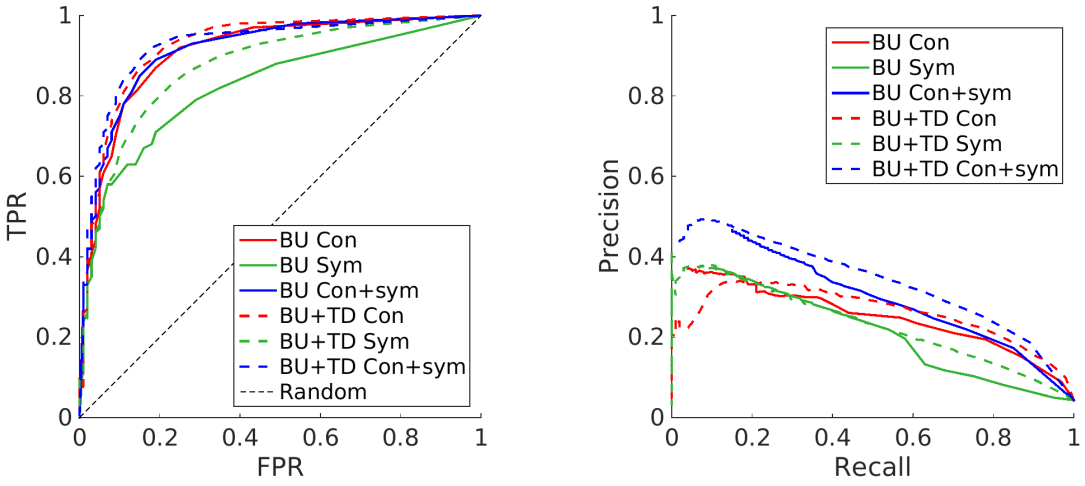


Figure 4.28: Performance of the defect detector using the Bayesian framework SUN: (left) ROC curves and (right) PR curves. Performance curves obtained for the six configurations of the framework, by variation of the saliency threshold τ_S. BU refers that the detector uses only bottom-up information, while BU+TD means that also combines top-down data.

incorporated. In both cases, the three versions of the defect detector have been considered: using only contrast information, using only symmetry information, and using both features together.

As already happened with the generic framework, the contrast-based detectors perform better than the symmetry-based ones, and the detectors combining both features provide the best results: their ROC curves are slightly above the curves obtained using only contrast, and the precision values are considerably higher, especially for low recall values. The AUC values for the six detectors using the Bayesian framework can be found in Table. 4.8.

Regarding the incorporation of top-down knowledge, this increases the performance of the detector regardless of the feature/s employed. Looking at the ROC curves provided in Fig. 4.28 [left], the higher improvement is obtained for the case of only using symmetry. In terms of AUC, the incorporation of top-down knowledge to the symmetry based detector increases this metric by 5.95%, while the improvement is only about 1.76% and 1.20% when respectively using, only contrast, and both features together.

Finally, we have compared the performances of the generic and the Bayesian frameworks. Figure 4.29 provides the ROC and PR curves corresponding to the single-feature detectors (i.e. using only contrast or symmetry), together with the combined detectors which provide the best

Table 4.8: AUC values for the defect detector using the Bayesian framework SUN. These values correspond to the ROC curves presented in Fig. 4.28.

	Contrast	**Symmetry**	**Con.+sym.**
Bottom-up	0.908	0.824	0.914
Bottom-up + top-down	0.924	0.873	0.925

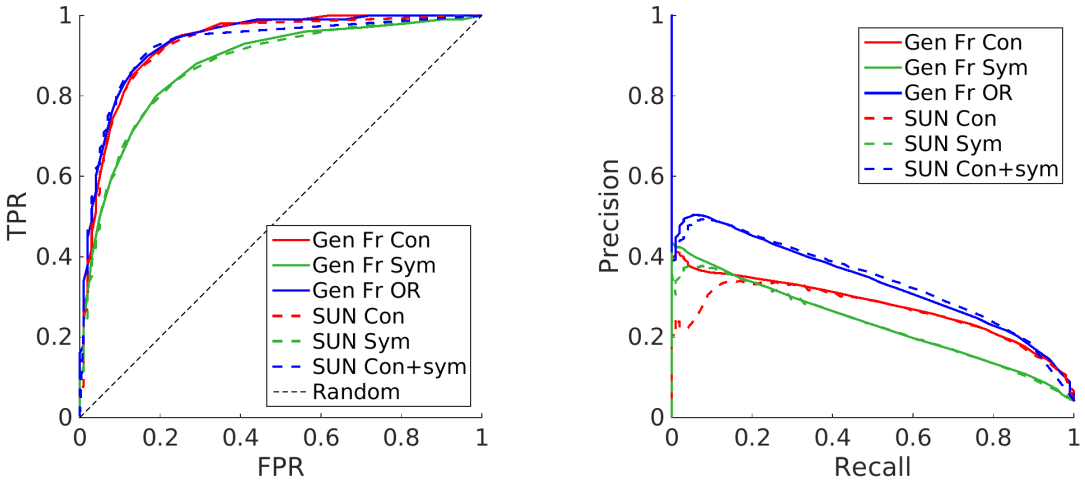


Figure 4.29: Comparison of the performance of the generic and Bayesian (SUN) defect defection frameworks: (left) ROC curves and (right) PR curves. The OR combination has been selected to merge the contrast and symmetry information when using the generic framework. SUN curves are obtained merging bottom-up and top-down information.

performance. The OR combination has been selected among the different combinations which permit the generic framework, since this provides a higher AUC than the AND combination (see Table 4.7) and it is conceptually simpler than the ORA combination. Regarding the detectors using the Bayesian framework, Fig. 4.29 plots the values corresponding to the use of both top-down and bottom-up information. The results show that both frameworks provide a very similar performance.

By way of example, Fig. 4.30 shows some saliency maps produced by the different general defect detectors considered in this section. As can be observed, all they tend to label in lighter gray those areas which are labelled as defective in the ground truth image. Likewise, Fig. 4.31 presents some final detection outputs (i.e. after applying the threshold τ_S) obtained using the OR combination operator.

The execution times have also been measured for the different frameworks and configurations. These can be found in Table 4.9. The values show that computing contrast takes much less time than computing symmetry: considering an image with 1024×768 pixels, computing contrast takes around 0.5 s, while computing symmetry takes almost 5 s. As can be expected, the versions that combine both features require a bit more time. Regarding the Bayesian framework, there is no penalty for introducing top-down knowledge, in comparison with the

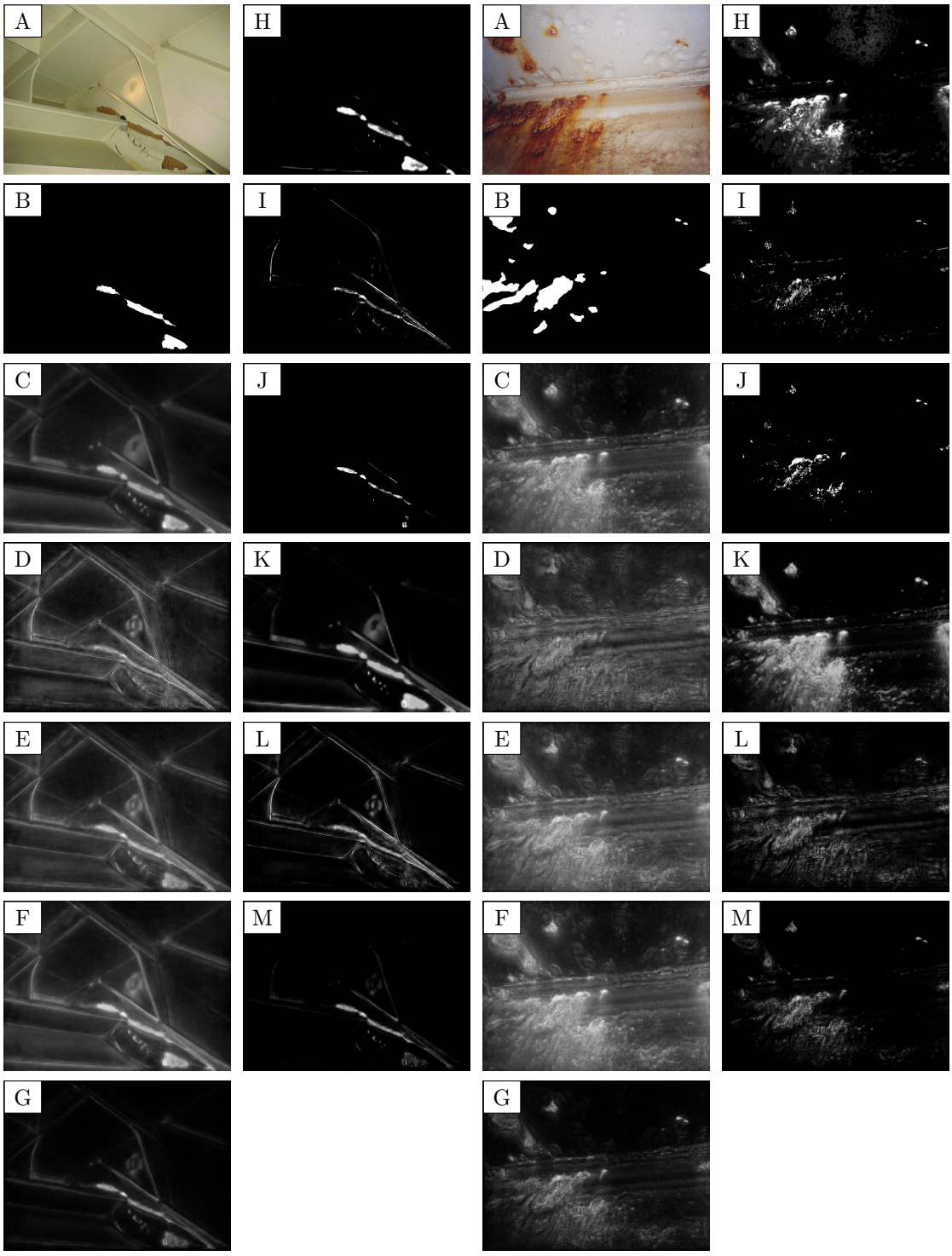


Figure 4.30: Saliency maps obtained using the different methods: (A) input image, (B) ground truth, (C-G) outputs for the bottom-up generic framework using contrast (C), symmetry (D), and the operators OR (E), ORA (F) and AND (G), (H-M) outputs for the Bayesian framework using bottom-up data about contrast (H), symmetry (I) and both features (J), and merging also top-down data from contrast (K), symmetry (L) and both features (M).

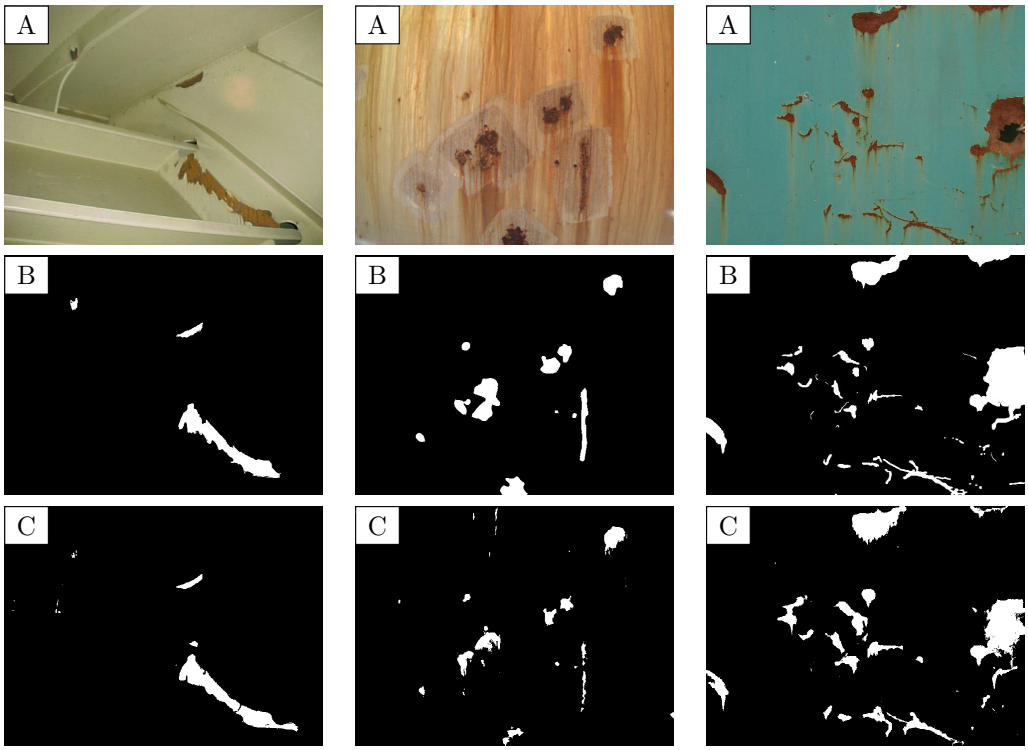


Figure 4.31: General defect detection results obtained using the OR combination operator: (A) input image, (B) ground truth, (C) defect detector output.

version which only makes use of bottom-up data.

4.5.6 Conclusions

Unlike previous sections, where different algorithms have been presented for detecting specific defects on vessel structures (i.e. corrosion or cracks), in this section we have dealt with the detection of generic defects. To face this problem we have focused on the idea of saliency. In this regard, contrast and symmetry have been selected as texture-based saliency-related features to detect defective areas in pictures taken from vessel hulls.

Two different frameworks have been described to compute the conspicuousness of the areas in the image, based on, at least, one of the two selected features, contrast and symmetry. On the one hand, a generic framework has been presented to provide an easy integration of different features and their combinations. Apart from the single-based detectors, three combination operators have been proposed to combine contrast and symmetry information. On the other hand, a Bayesian framework has been applied to the vessel inspection problem. This framework makes use of PDFs computed in a previous training stage, to estimate in probabilistic terms how salient is a given pixel according to the value of the selected features. Furthermore, the Bayesian framework allows the introduction of top-down knowledge to enrich the information provided by bottom-up saliency.

Table 4.9: Execution times of the different general defect detectors.

	Generic bottom-up framework					**Bayesian framework**					
						Bottom-up			**Bottom-up + top-down**		
	Co.	**Sy.**	**OR**	**ORA**	**AND**	**Co.**	**Sy.**	**Both**	**Co.**	**Sy.**	**Both**
μs/pixel	0.62	6.11	6.58	6.64	6.64	0.73	6.25	6.64	0.76	6.28	6.66
s/img.*	0.49	4.81	5.17	5.22	5.22	0.57	4.92	5.22	0.59	4.94	5.24

Co. and Sy. refer to contrast and symmetry features respectively.
* considering a 1024×768 image.

The use of contrast and symmetry features has shown to provide good detection results with both frameworks and with the different combinations evaluated. The results obtained with the generic and Bayesian frameworks (when introducing top-down data) are very similar. Because of that, the generic framework is preferable since it does not require a previous training stage.

Regarding the features, contrast provides considerably better results than symmetry for the images in the dataset. Nevertheless, all the different combinations evaluated outperform the contrast-based detectors, providing higher precision values (see Fig. 4.29 [right]), what justifies the introduction of symmetry information.

4.6 Combination of Saliency and Specific Defect Search for Boosted Defect Detection

This section discusses the idea of improving the performance of the corrosion and crack detectors by means of previously filtering the images using a saliency-based generic defect detector. In other words, the high detection performance presented by the different versions of the generic defect detector (see Section 4.5.5) is used here to boost the performance of the specific defect detectors.

The idea of a cascade of classifiers has been previously used in this dissertation. Firstly, this methodology has been used in Section 4.3 to build the corrosion detector, where a colour and a texture-based classifiers are chained to progressively filter the candidate pixels. In second place, the entire corrosion detector has been used to guide the crack detector in Section 4.4, where only those pixels labelled as corroded are used to start the region-growing procedure performed within the crack detector.

To improve the performance of the corrosion and crack detectors, the generic OR-based defect detector is selected, in this section, as first stage for detecting both corrosion and cracks. Among the different methods evaluated in Section 4.5, the combination of contrast and symmetry information using the OR operator has led to higher precision values than

the single-feature detectors (see Fig. 4.27 [right]), it outperforms the AND combination in terms of AUC, and it is conceptually simpler than the ORA combination, which separates the contrast information into three different channels.

On this occasion, the classification threshold for saliency τ_S has to be set to the appropriate value, in order to maximize the performance of the subsequent classifier, i.e. the corrosion or crack detector. Two different strategies are followed:

- To filter the input provided to the specific corrosion detector, the threshold τ_S is set to lead to a very high TPR, despite this means to move the detector a bit further from the (0, 1) corner in ROC space, increasing the FPR. The idea is to provide all the corroded pixels (positive pixels) as input for the corrosion detector, i.e. keep the FN close to zero.

- To filter the input provided to the crack detector, τ_S is set so as to ensure that a moderate collection of pixels will be available as seeds for the region-growing procedure. In order to make certain that these pixels actually belong to a crack, the threshold is set to provide a very reduced FPR, despite this also means a limited TPR. Notice that this strategy has also been applied to configure the corrosion detector that guides the crack detector in Section 4.4.2.

4.6.1 Boosted Corrosion Detector

Among the different corrosion detectors presented in Section 4.3, we have selected the one which provides the best performance. This corresponds to the main approach which makes use of RGB local stacked histograms and the GLCM energy. Notice that this method has also been selected to guide the crack detector of Section 4.4.

Figure 4.32 shows the ROC curve produced by the corrosion detector together with the (FPR, TPR) point corresponding to the saliency-based generic defect detector, once the threshold τ_S has been configured as indicated before. As can be observed, this point is situated in the coordinates (0.31, 0.97) of the ROC space, which is above the curve provided by the original corrosion detector.

Notice that, if the output of the generic defect detector is provided as input to the corrosion detector, the new ROC curve resulting from varying its parameters will end up at point (0.31, 0.97), which corresponds to the corrosion detector that labels as positive all the pixels that have passed the first classifier. Therefore, we can expect that the new ROC curve will pass above the original curve.

Figure 4.33 compares the performance of the original corrosion detector with the new version boosted by the saliency-based defect detector. As expected, the results show that the boosted version outperforms the original detector: on the one hand, the new ROC curve is closer to the (0, 1) corner; on the other hand, the boosted detector leads to higher values of precision, as shown in Fig. 4.33 [right].

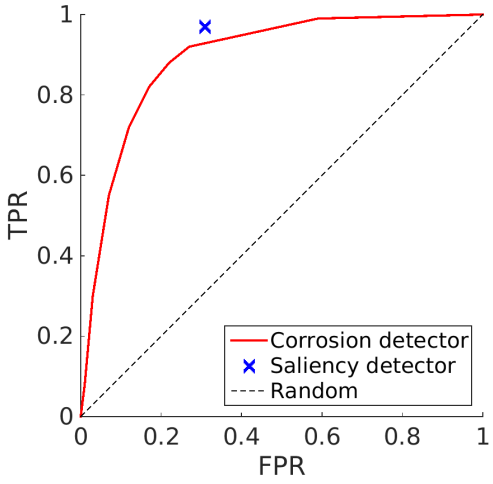


Figure 4.32: Comparison of the performance of the corrosion and the saliency-based generic defect detectors.

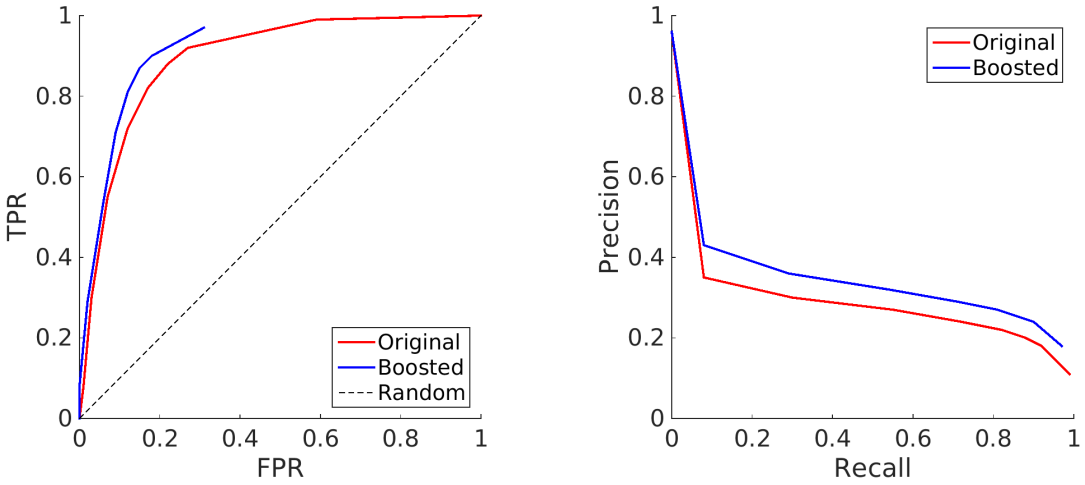


Figure 4.33: Performance improvement of the saliency-boosted corrosion detector with regard to the original method: (left) ROC curves, (right) PR curves. Performance curves obtained for energy threshold $\tau_E = 0.4$, and by variation of threshold τ_D.

By way of example, Fig. 4.34 compares the output provided by the original and boosted versions of the corrosion detector using the same configuration of their parameters ($\tau_E = 0.4$ and $\tau_D = 0.175$). As can be observed, the boosted version presents some less false positives than the original version.

4.6.2 Boosted Crack Detector

To boost the performance of the crack detector, we set the OR defect detector before the original crack detector, similarly to what we have done when using the corrosion detector to

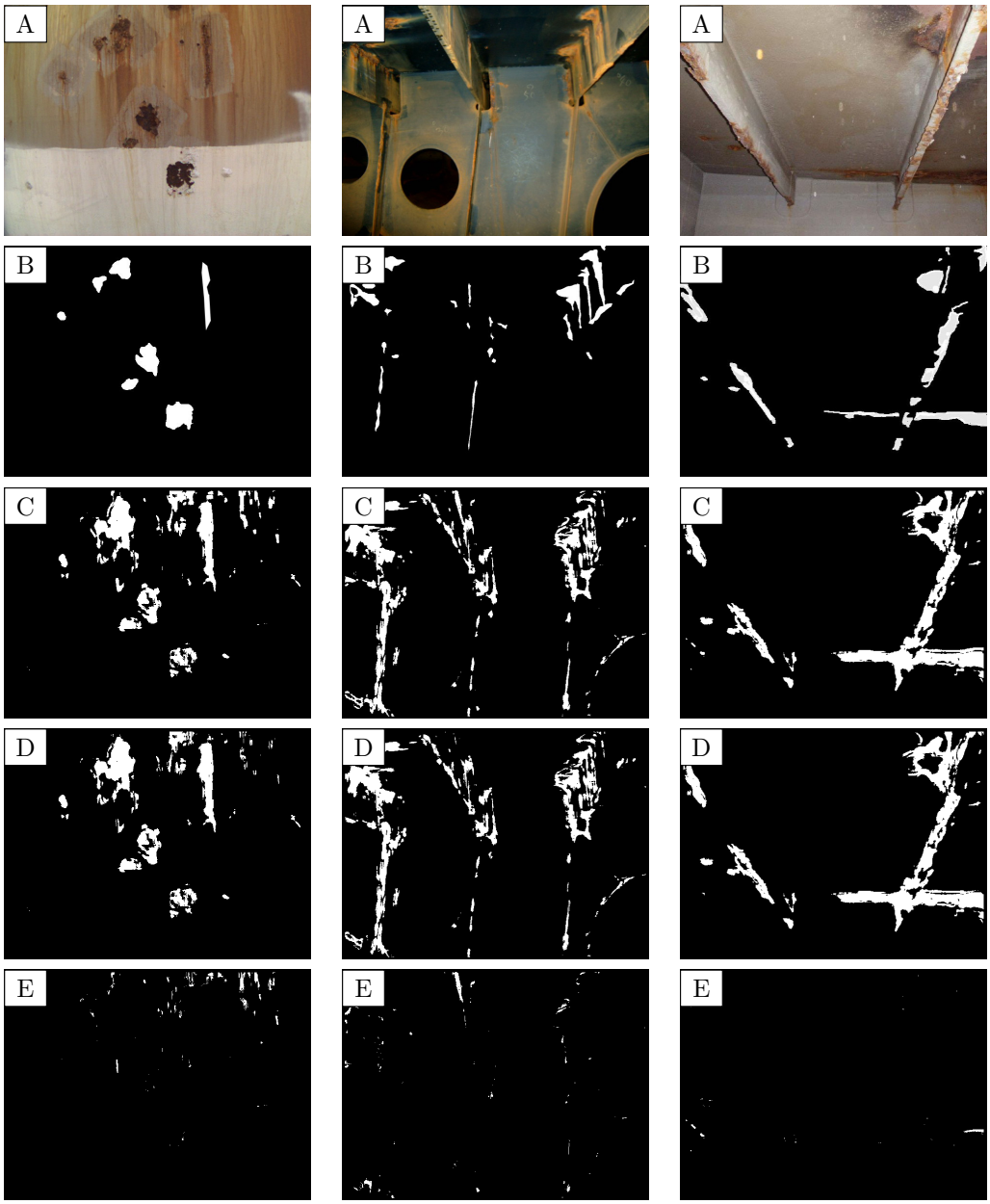


Figure 4.34: Corrosion detection results for some images of the dataset: (A) input image, (B) ground truth, (C) output provided by the original version of the method, (D) output provided by the saliency-boosted version, and (E) difference image showing those areas that are labelled as corrosion by the original version but discarded by the saliency-boosted version.

guide the crack detector in Section 4.4.2. As previously indicated, the threshold τ_S of the saliency-based method is set to provide a very reduced FPR. The selected value results in a detector whose performance is situated at the (0.05, 0.65) point of the ROC space. These metrics indicate that 65% of the pixels belonging to some crack are identified by the generic defect detector, while it results in just a few false positives.

Table 4.10: Performance improvement of the saliency-boosted crack detector with regard to the original and corrosion-guided methods

	Original	**Corrosion-guided**	**Saliency-boosted**
Precision	0.60	0.75	0.74
Recall	1	0.84	0.92

Table 4.11: Performance improvement of the saliency-boosted crack detector with regard to the original and corrosion-guided methods. Results obtained after removing 4 images of the dataset.

	Original	**Corrosion-guided**	**Saliency-boosted**
Precision	0.74	0.87	0.90
Recall	1	1	0.95

The pixels provided by the saliency-based defect detector are candidates to start a region growing procedure. As indicated in Section 4.4, this is performed only if the pixel is over an edge and if it is darker than its neighbourhood.

Table 4.10 compares the precision and recall metrics obtained for the original, corrosion-guided and saliency-boosted versions of the crack detector. As can be observed, the boosted method provides a precision considerably above the one obtained with the original method, and similar to the one provided by the corrosion-guided method. Nevertheless, the recall value is not so low as when guiding the inspection with corrosion.

For completeness, and to provide an additional comparative evaluation, Table 4.11 reports the performance metrics obtained for the reduced dataset which results from removing the four images that include cracks which are not affected by corrosion (see Section 4.4). The precision obtained with the saliency-boosted method is 0.9, which is the highest among the three versions. Nevertheless, the recall value does not reach 100%, since 5% of the cracks are not detected.

Figure 4.35 compares the results provided by the three detectors proposed for some images of the dataset. The results show that the saliency-boosted version leads to less false positive detections than the other versions. The image in the middle is one of the four images where the corrosion-guided method fails since the crack does not coincide with corrosion. Both the original and the saliency-boosted versions are able to identify the crack. On the contrary, the right image shows two cracks which are affected by corrosion, for which the original and the corrosion-guided versions are able to detect both cracks, but the saliency-boosted version is not, i.e. it misses the crack on the left side of the image. This is probably because the crack is located in a dark and roughly-textured area of the image, so that it does not result salient to the OR method.

4.6.3 Conclusions

In this section, a generic defect detector based on saliency has been evaluated to guide the corrosion and crack defect detectors. The classifiers are combined in a cascade, where the generic defect detector is used in the first stage to filter the pixels of the input image.

In the light of the results obtained using the combination, we can state that the saliency-based classifier allows boosting the performance of both specific defect detectors, reducing their false positive detections and, thus, increasing their precision (see Fig. 4.33 [right] and Table 4.10).

To finish, it is interesting to point out that the use of saliency may cause some defects to be misclassified as false negatives. This could occur, for example, in an image where the pixels are affected by corrosion. In this case, the saliency-based detector will fail indicating just the most conspicuous areas, despite the corrosion detector alone could be able to identify the entire image. Nevertheless, notice that this scenario does not match the idea described in Section 4.5, where vessel structures are described as large and uniformly textured surfaces and where defects are considered a rare phenomena.

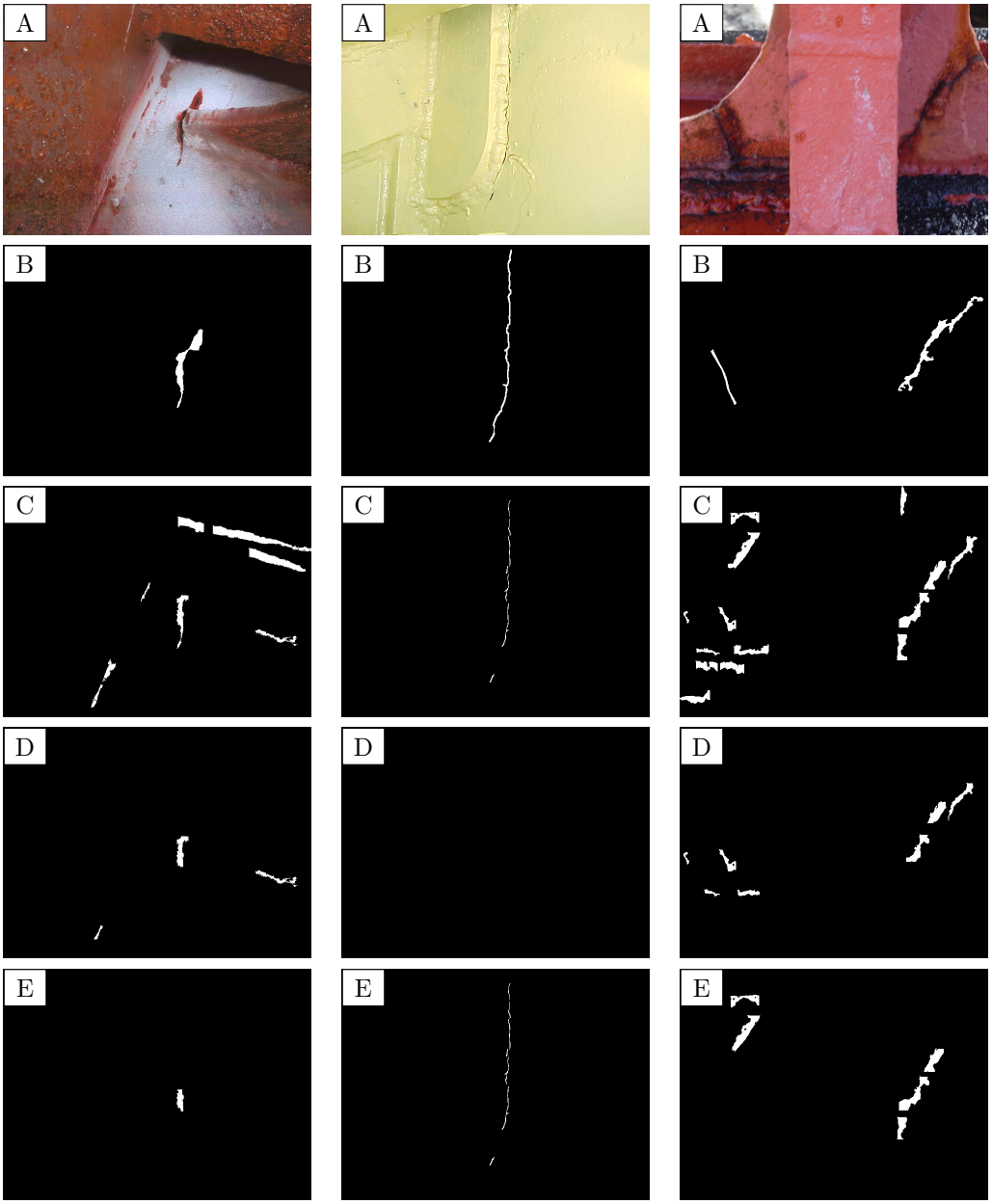


Figure 4.35: Crack detection results for some images of the dataset: (A) input image, (B) ground truth, (C-E) crack detection using, respectively, the original version, the corrosion-guided version, and the saliency-boosted version.

Field Trials Results

This chapter reports on the experimental results obtained using the technological tools described along this dissertation for the inspection of a real vessel. The first test campaign was performed on November 2015 on board an Aframax oil tanker, with a capacity above 100000 dwt and whose size is 237.64 m (length)×42 m (breadth)×21.3 m (height). During these field tests, preliminary results were obtained since the MAV and the vision-based defect detectors were not yet fully developed. Nevertheless, useful feedback was obtained for the design process of such tools. For confidentiality reasons, an image of this vessel can not be included in this dissertation. Nevertheless, Fig. 5.1 shows an oil tanker similar to the vessel visited, as well as some pictures taken during these field tests inside one of its cargo holds.

In this chapter we focus on the experimental results obtained during a second inspection campaign conducted in May 2016, when the technology tools were fully operative. In Section 5.1, the vessel is described, detailing the different compartments where the experiments were performed. Section 5.2 addresses the experiments involving the MAV described in Chapter 3, detailing the sensor suite employed and providing results for the different compartments considered. Section 5.3 reports the defect detection results obtained when using the methods proposed in Chapter 4 to analyse the images taken by the MAV. Among all the proposed approaches, this section focuses on the defect detection methods that have provided the best results. Finally, Section 5.4 draws some conclusions about the results obtained during field trials.

5.1 Testing Facilities

The field trials were performed on board a Handymax bulk carrier with dwt above 45000 tons, and whose size is 190 m (length)× 32 m (breadth)×16.5 m (height). A picture of this vessel can not be included for confidentiality reasons (as before), but Fig. 5.2 provides a general view and the plans corresponding to a vessel with the same characteristics.

During the test campaign, the MAV was operated in three different compartments:

- cargo hold #4, in front of hull frames #75-90 (see Fig. 5.3 [A]),
- water ballast topside tank #3, in front of frames #111-131 (see Fig. 5.3 [B]), and

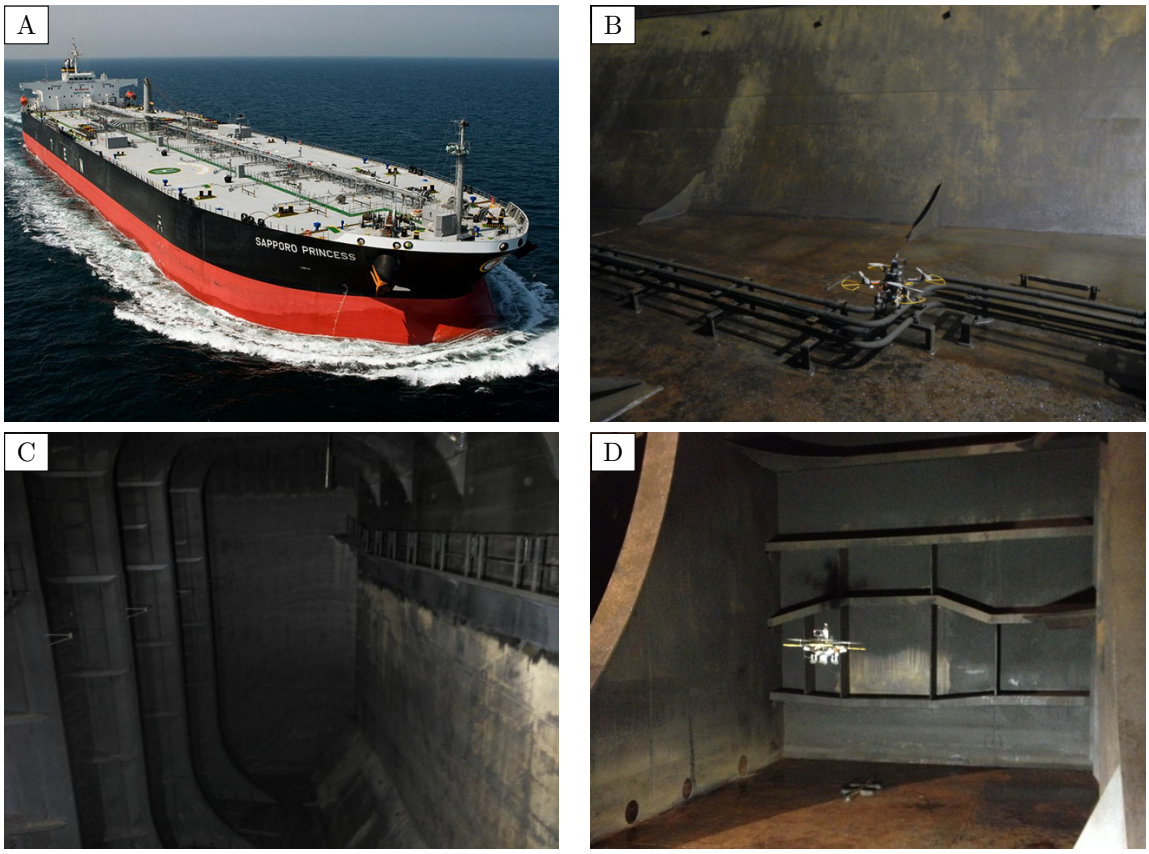


Figure 5.1: Images corresponding to the first field trials: (A) Aframax oil tanker similar to the vessel visited, (B-D) some pictures taken performing tests inside a cargo hold.

- the forepeak tank, between frames #215 and #225 (see Fig. 5.3 [C]).

The operating conditions in each compartment were very different. On the one hand, the metallic plates inside the cargo hold did not exhibit much corrosion since the state of their coating was good. On the other hand, the metallic surfaces inside the topside and forepeak tanks presented several corroded areas due to the water which is introduced to ballast the vessel.

Regarding the illumination conditions, the light inside the cargo hold could be relatively adjusted, since the hatch could be opened and closed. On the contrary, the forepeak and topside ballast tanks were dark spaces accessible through a manhole-sized entry point (see Fig. 5.4 [right]); artificial lights had to be used to allow for a proper visual inspection.

5.2 Experiments using the Aerial Platform

The MAV used during the field trials was the AscTec Pelican platform equipped with the SS2. This sensor suite, based on the use of a laser scanner, is suitable for flying in dark spaces (e.g.

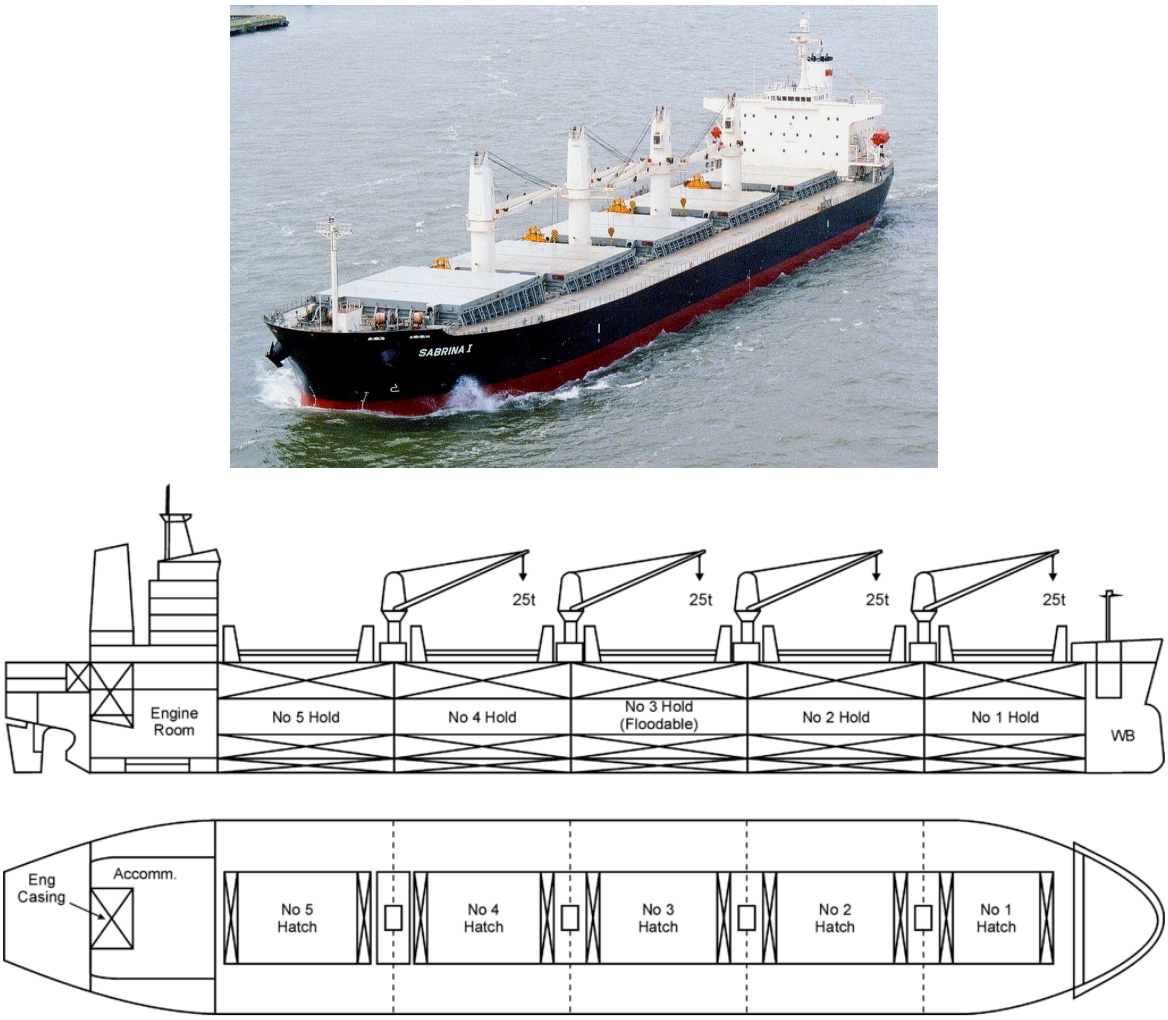


Figure 5.2: Handymax bulk carrier similar to the vessel visited during the field tests. Diagram by Rémi Kaupp taken from Wikipedia [1]

inside a ballast tank) where the optical flow sensors, employed in the SS1, can not operate (see Section 3.6.1 for a detailed description of the aerial platform equipment). Furthermore, this vehicle is equipped with a high power LED to illuminate the inspected surface if necessary. Figure 5.4 [left] shows the MAV inside the cargo hold, while Fig. 5.4 [right] shows the vehicle being introduced into the topside ballast tank through a manhole.

All the experiments have been performed following the same procedure:

1. the vehicle is situated in a flat and obstacle-free area for the take-off,

2. the user sends the take-off command using a gamepad/joystick and the vehicle starts the flight,

[1] https://en.wikipedia.org/wiki/Bulk_carrier

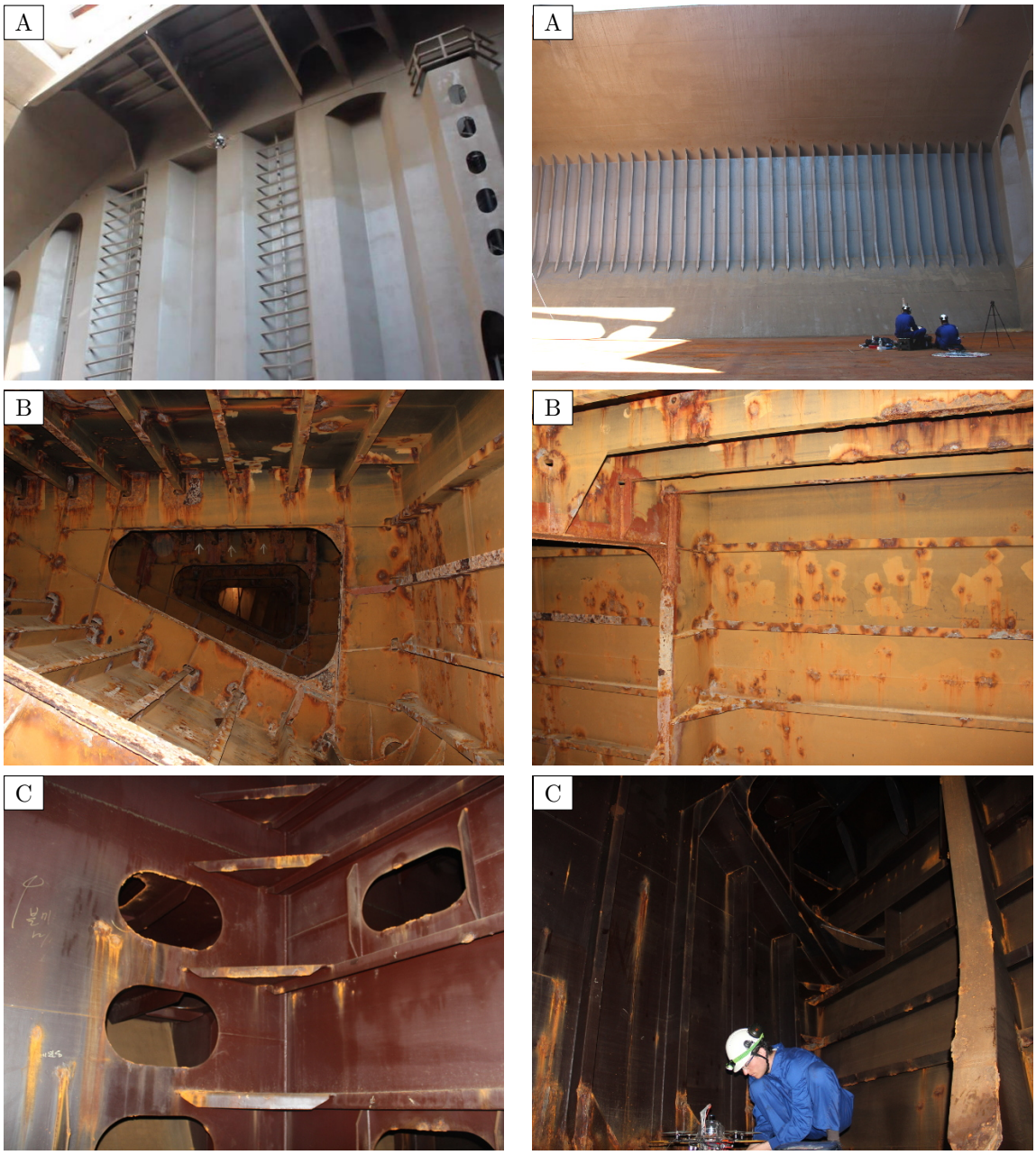


Figure 5.3: Compartments inspected during the field trials on board the bulk carrier: (A) cargo hold, (B) topside tank and (C) forepeak tank.

3. the user approximates the platform to the area where has to take place the inspection, while the architecture based on SA prevents possible collisions with the vessel structures,

4. the user can optionally enable the *inspection mode* to make the vehicle move smoothly and keep at a constant distance to the inspected surface,

5. a sequence of pictures can be started when desired,

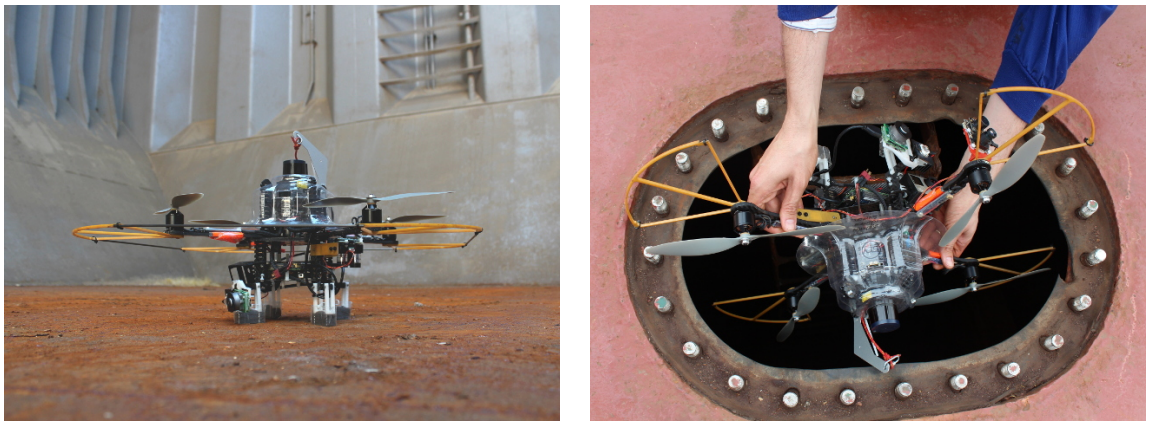


Figure 5.4: MAV used for the field trials: (left) inside the cargo hold, (right) being introduced into the topside tank through a manhole.

6. the user can command the platform along the Y and Z axis (also X if the *inspection mode* is not enabled) to perform the inspection,
7. the sequence of pictures can be stopped when desired,
8. the *inspection mode* is disabled (if it was enabled),
9. the user commands the platform to an obstacle-free area for landing, and
10. the user indicates the command for landing.

Figure 5.5 shows some pictures taken during testing at the cargo hold. In a first session, flights took place in front of the aft bulkhead, frames #75-78, while in a second session, testing focused on the cargo web frames, starboard side, frames #78-90.

Before performing the tests, the base station was placed in front of the respective operating area. Then, tests were conducted to determine the correct behaviour of the platform in the environment. To this end, sensors and control architecture outputs were first checked with the vehicle at the base station, not flying. Next, tests were conducted in front of the different surfaces progressively attaining larger altitude as testing progressed. Figure 5.5 [B] shows the platform during some of the flights that were performed in the cargo hold.

By way of illustration, Fig. 5.6 shows the paths estimated for three of these flights. In the three cases, paths were successfully estimated by means of the GMapping SLAM method, which makes use of the data provided by the laser scanner. Figure 5.7 shows a longer flight in front of a large wall, in which the robot flew from left to right and then back. On this occasion, the SLAM module got confused just before coming back, and, because of this, the path does not ends where it started. This error is probably due to the long distance to all the corners and other distinguishable structural elements inside the cargo hold. To show the actual path followed by the MAV, Fig. 5.7 [C-D] provides the first part of the flight, while

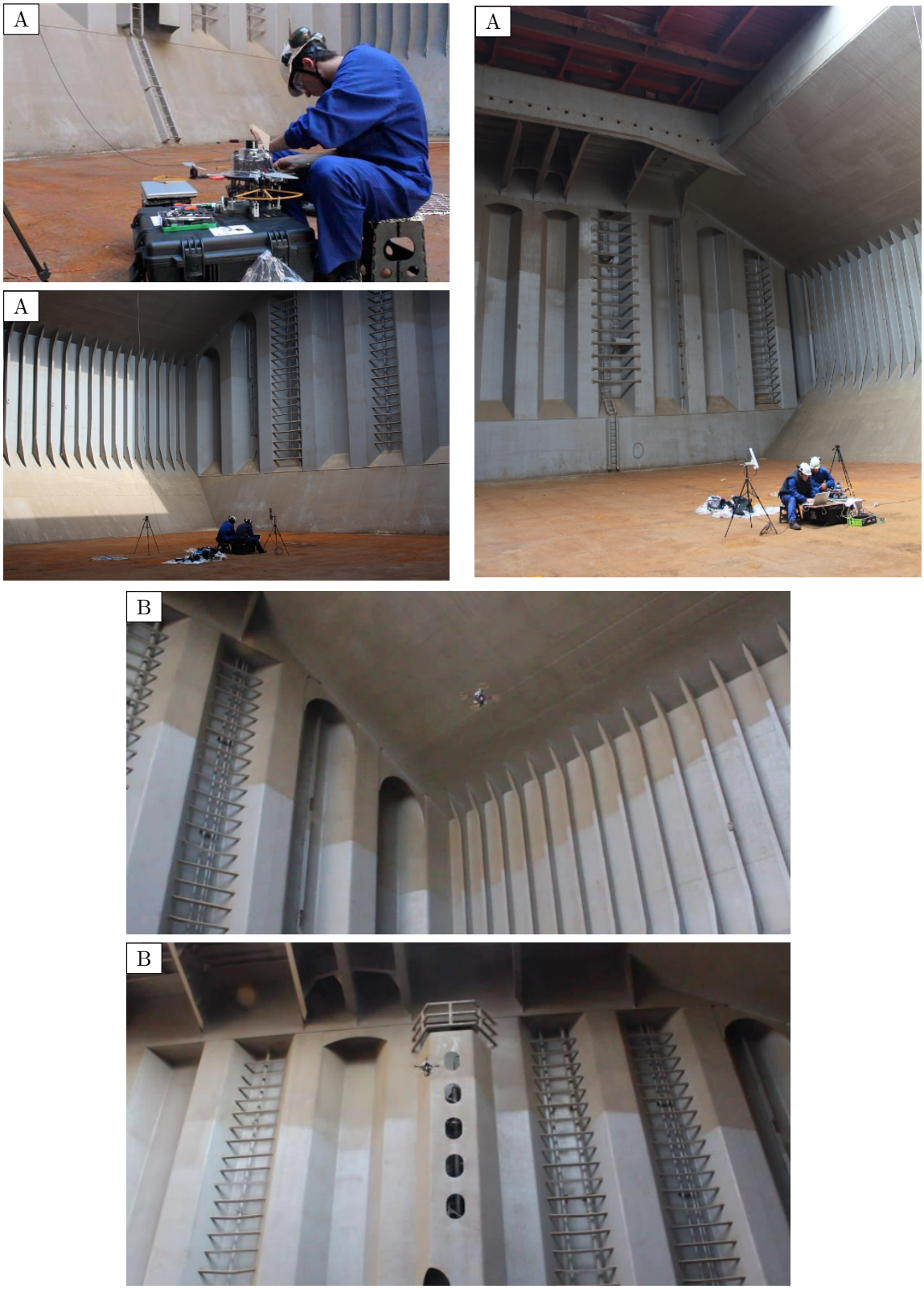


Figure 5.5: Some pictures to illustrate testing in the cargo hold: [A] personnel preparing the experiments, [B] the aerial platform in flight.

Fig. 5.7 [E-F] provides the second part. Some of the images captured by the on-board camera during this last flight can be found in Fig. 5.8.

Figure 5.9 shows some pictures of the experiments performed at the topside tank, which took place in front of frames #111-131. Unlike the cargo hold, this compartment is a confined space where the self-preservation capability, included in the SA framework, is highly required. These tests also allowed us to check the capability of the platform to take pictures under low-light (hatchway open) and under completely dark (hatchway closed) conditions. As for the previous area, first tests were conducted to determine the correct behaviour of the platform in the environment, checking sensors and control architecture outputs without flying. Figure 5.9 [B] shows the platform during one of the flights.

By way of illustration, Fig. 5.10 shows the paths estimated for two of the flights performed in the topside tank. In the first case, the vehicle was flying with some light available from the hatch. In the latter case, the hatch was closed, and hence the area was completely dark. In both cases, the SLAM method provided a successful position estimation. Some of the images captured by the on-board camera during this last flight can be found in Fig. 5.11. As can be observed, these are adequately illuminated thanks to the use of the high power LED installed in the MAV.

Finally, Fig. 5.12 shows some pictures of the experiments performed at the forepeak tank. Testing took place among frames #215-225 in the upper stringer. As on previous cases, first tests were conducted to determine the correct behaviour of the platform within the environment, checking sensors and control architecture outputs without flying. All the experiments in this compartment were performed in complete darkness. Figure 5.12 [B] shows the platform during one of the flights.

By way of illustration, Fig. 5.13 shows the paths estimated for two of the flights performed in the topside tank, while Fig. 5.14 provides some of the images captured, during the latter flight, using the on-board camera. As happened in the topside tank, the self-preservation capability of the platform ensured an effective and safe operation, while the the laser-based SLAM provided correct position estimations thanks to the well-structured environment. The pictures provided by the camera module were also good, thanks to the illumination supplied by the on-board LED.

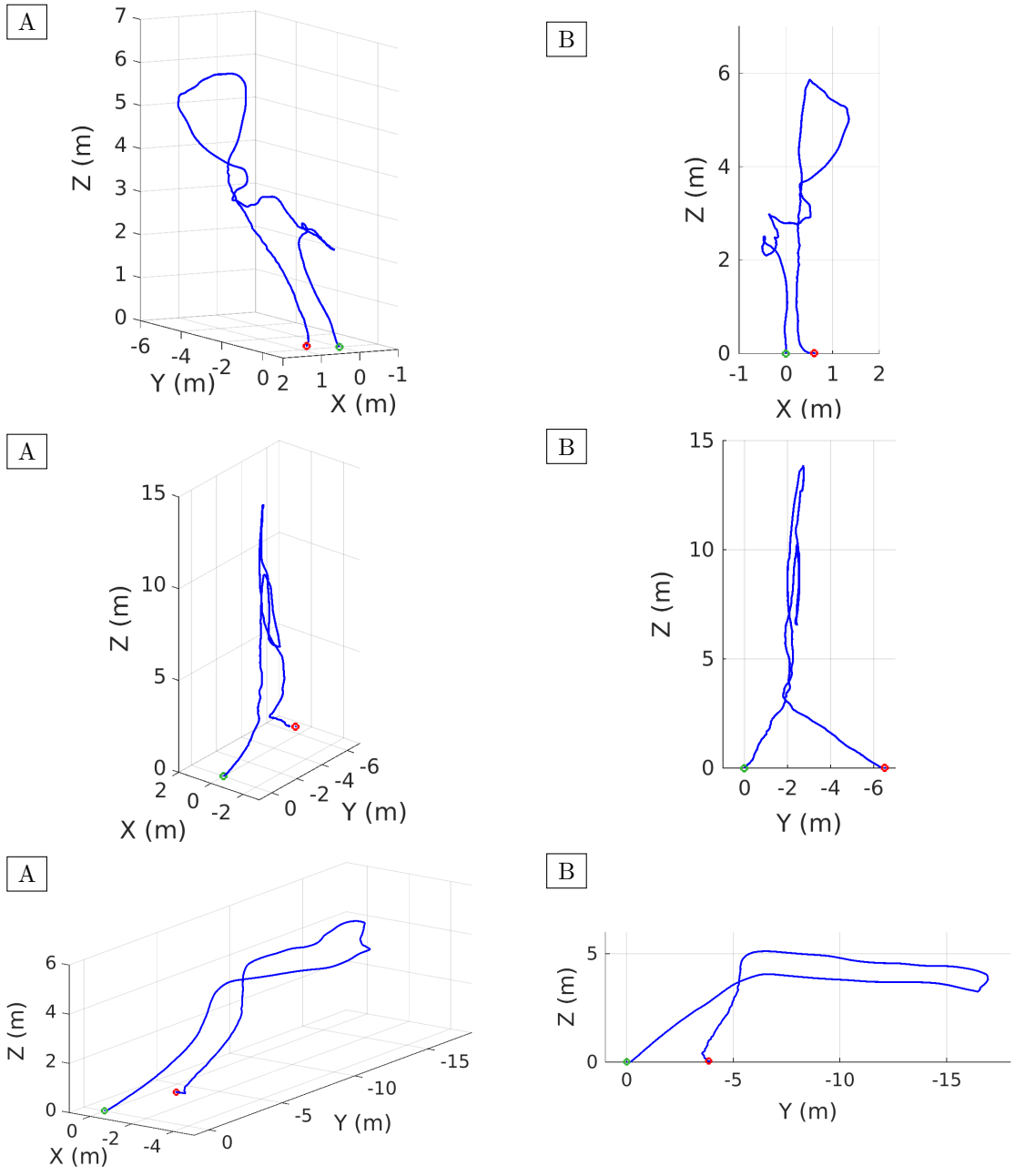


Figure 5.6: Estimated paths followed by the aerial robot during thee flights in the cargo hold: (A) 3D plot of the trajectories, (B) 2D projection of the trajectories. The green and red dots indicate the initial and final points respectively.

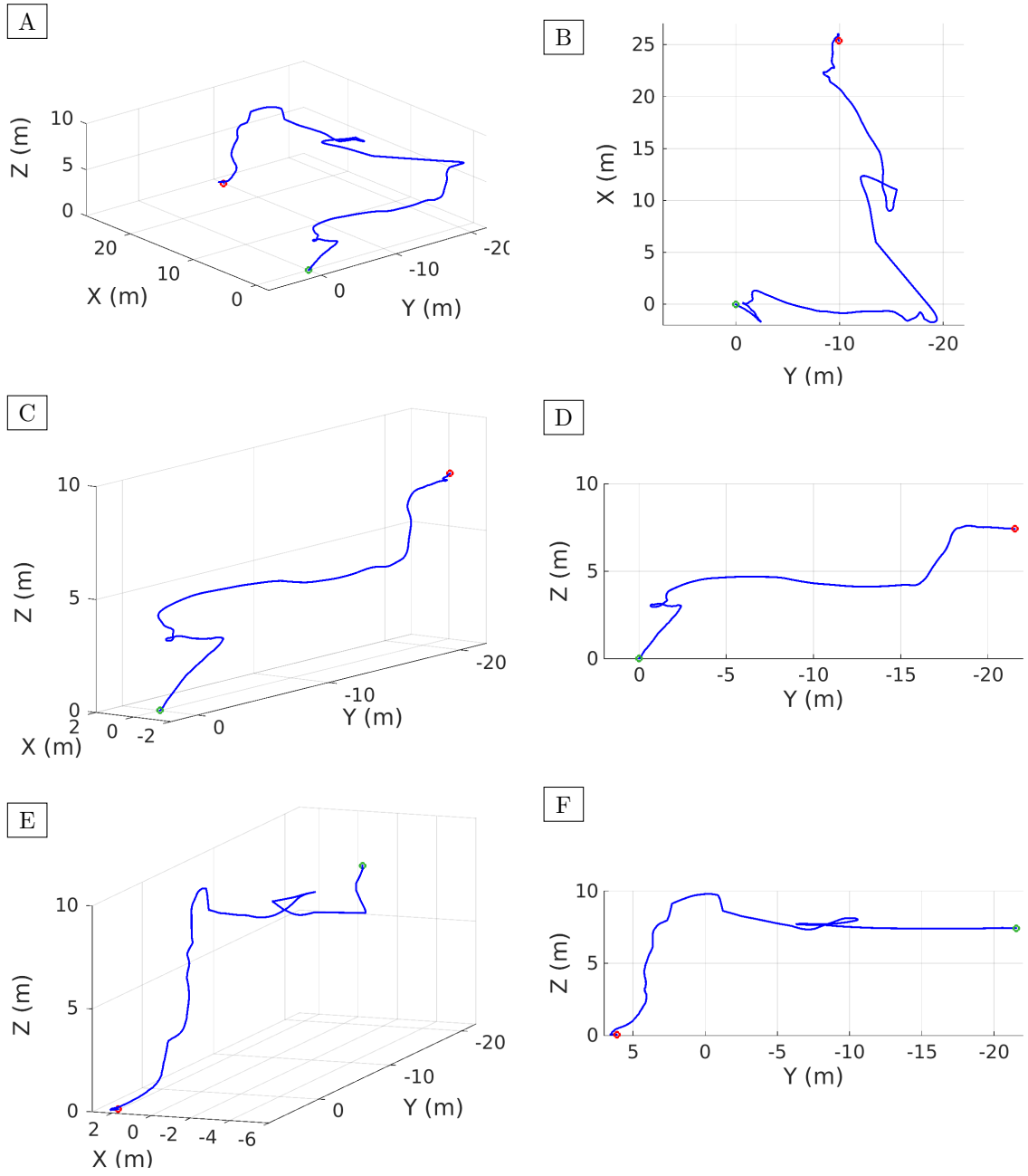


Figure 5.7: Erroneous estimation of the MAV path during a flight in the cargo hold: (A-B) 3D plot and 2D projection of the complete trajectory, (C-D) 3D plot and 2D projection of the first part of the flight, (E-F) 3D plot and 2D projection of the second part of the flight. The green and red dots indicate the initial and final points respectively.

Figure 5.8: Images taken with the on-board camera while flying in the cargo hold.

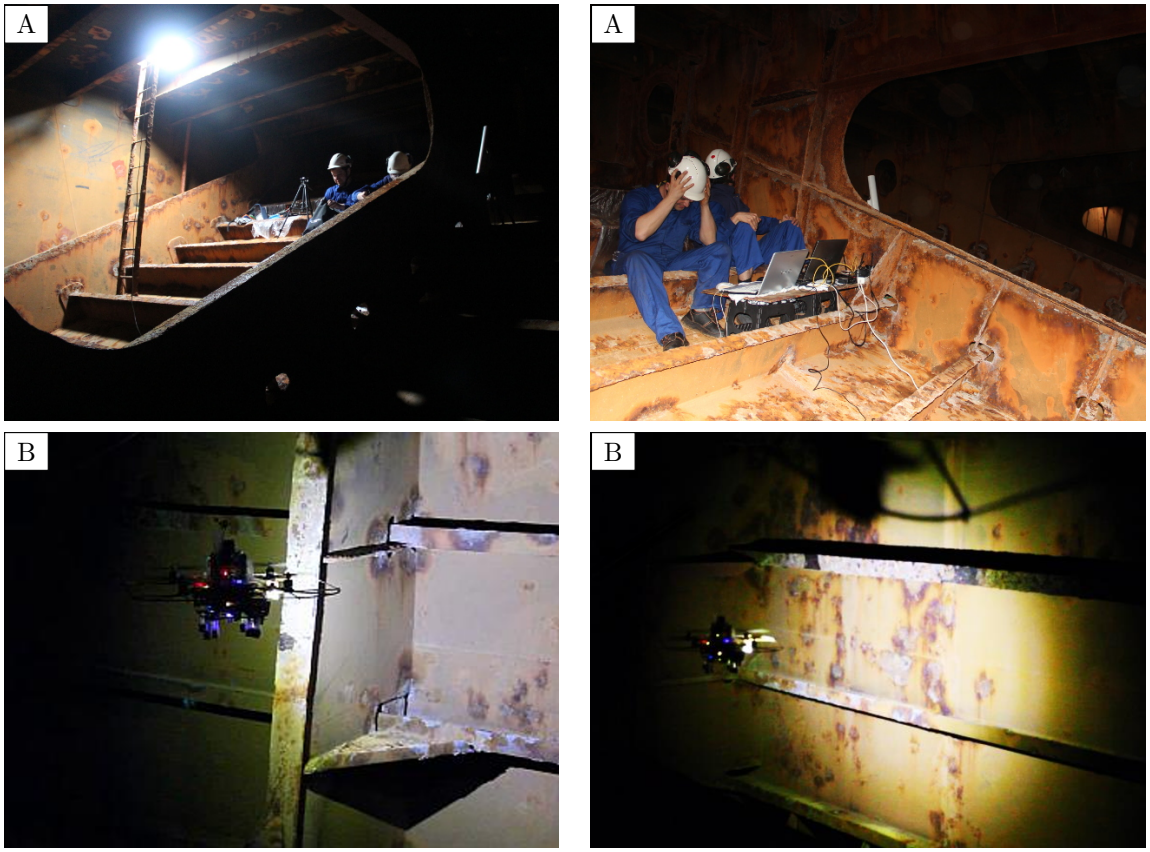


Figure 5.9: Some pictures to illustrate testing in the topside tank: [A] personnel preparing the experiments, [B] the aerial platform in flight.

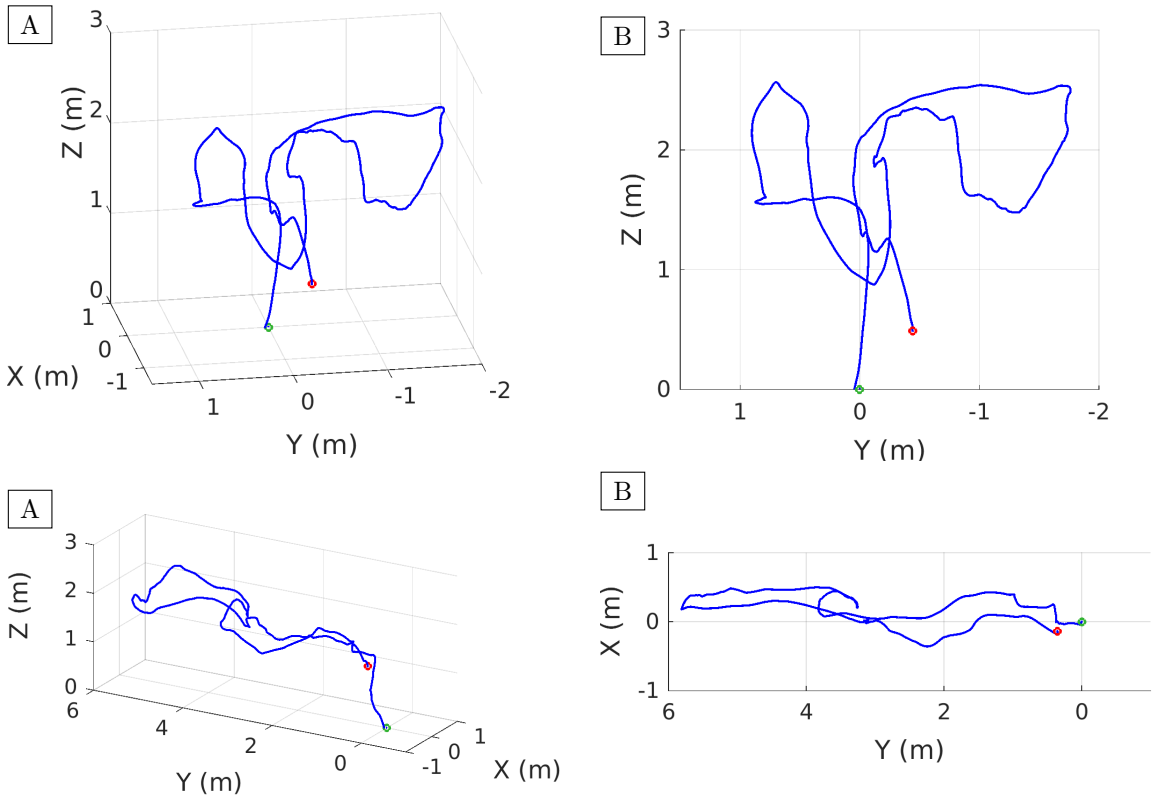


Figure 5.10: Estimated paths followed by the aerial robot during two flights in the topside tank: (A) 3D plot of the trajectories, (B) 2D projection of the trajectories. The green and red dots indicate the initial and final points respectively.

Figure 5.11: Images taken with the on-board camera while flying in the topside tank.

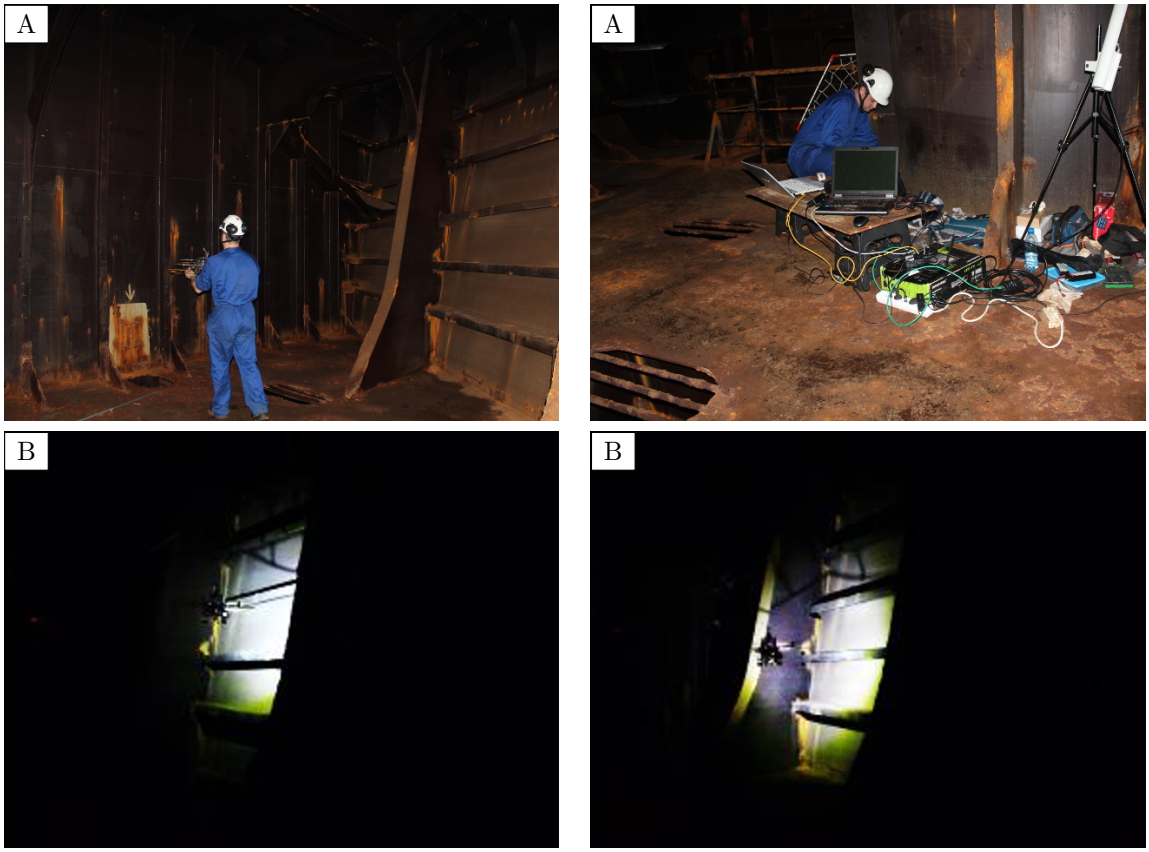


Figure 5.12: Some pictures to illustrate testing in the forepeak tank: [A] personnel preparing the experiments, [B] the aerial platform in flight.

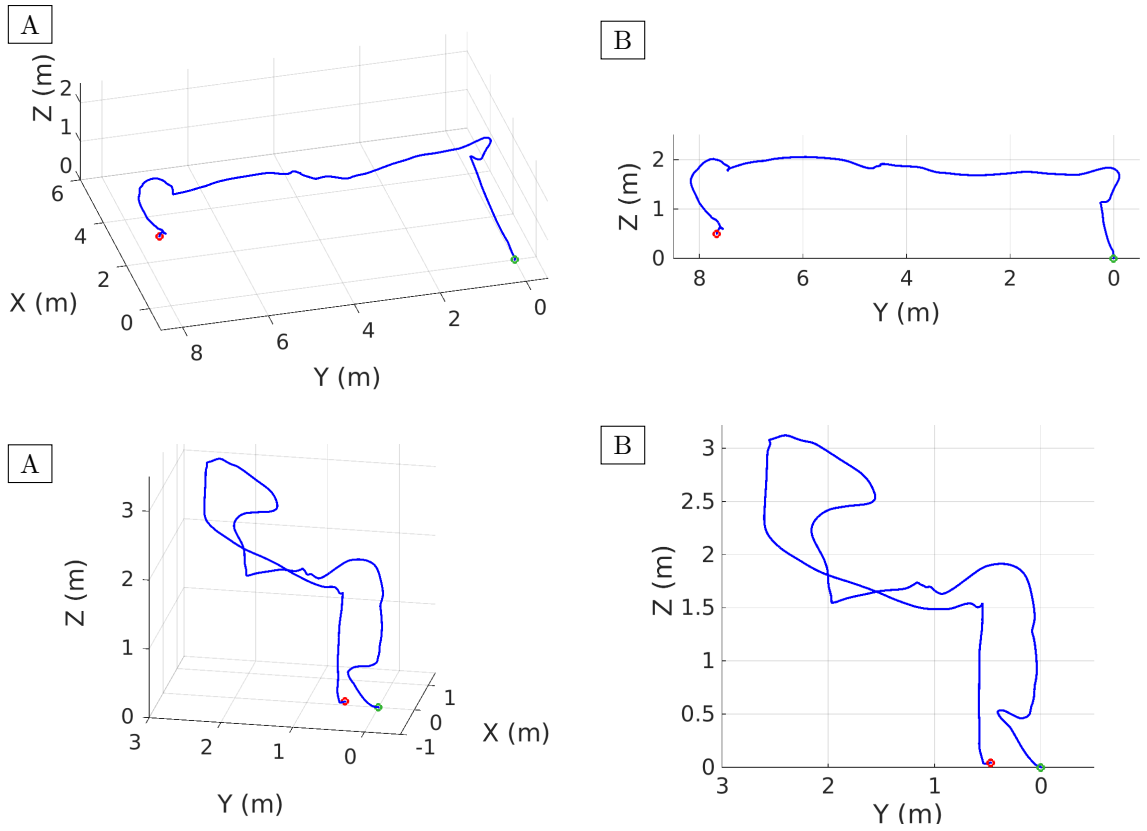


Figure 5.13: Estimated paths followed by the aerial robot during two flights in the forepeak tank: (A) 3D plot of the trajectories, (B) 2D projection of the trajectories. The green and red dots indicate the initial and final points respectively.

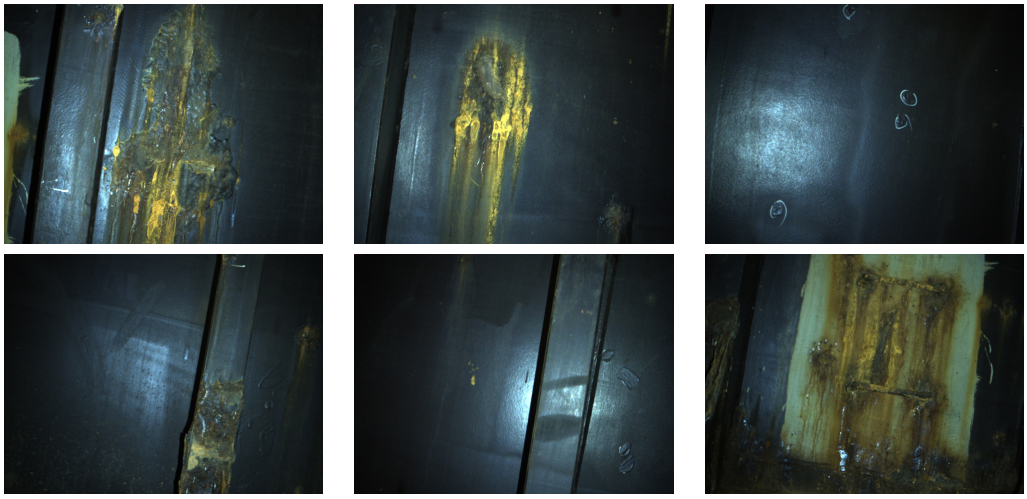


Figure 5.14: Images taken with the on-board camera while flying in the forepeak tank.

5.3 Defect Detection Experiments

To complete the visual inspection of the vessel compartments, the images taken with the MAV have been evaluated using the defect detectors presented in Chapter 4. Among them, we have used those which provided the best results during the performance evaluation, that is, the saliency-boosted defect detectors presented in Section 4.6. To be precise, just the corrosion defect detector has been of application since no cracks were observed in the vessel compartments visited during the testing campaign.

To evaluate the performance of the corrosion detector with the images taken by the MAV, three datasets have been generated, one for each of the compartments considered. The dataset comprising images taken in the cargo hold includes 79 pictures, the topside tank dataset comprises 83 pictures, and the dataset corresponding to the experiments performed in the forepeak tank includes 58 pictures. The ground truth for all these images has been generated following the same procedure described in Section 4.2. Figure 5.15 shows some of the images together with their ground truth. Notice that the images from the cargo hold dataset do no exhibit any corroded area.

As indicated in Section 4.6, the saliency-boosted corrosion detector results from the combination of the saliency-based bottom-up general defect detector, which merges contrast and symmetry information within a generic framework using the OR operator (see Section 4.5.3 for details), and the corrosion detector which makes use of GLCM energy and RGB local stacked histograms (see Section 4.3.6.1). The same configuration described in Section 4.6.1 of this combined method is used now to detect corrosion in the images taken using the MAV.

Prior to providing the detection results obtained with the boosted detector, we proceed to analyse the behaviour of its first stage, that is, the saliency-based general defect detector. Section 4.5.5 already provided a comparative evaluation of the results obtained with the two single-feature versions of the method, i.e. using only contrast or symmetry, and three alternatives which combined the information conveyed by both features using the operators OR, ORA and AND. Among them, the OR combination was selected to be used within the saliency-boosted corrosion detector. Nevertheless, it is interesting to re-evaluate the five versions of the detector to show how contrast and symmetry behave with the images taken from the different compartments of the vessel, whose corrosion looks quite different (see Fig. 5.15).

In this regard, Fig, 5.16 shows the performance metrics obtained for the different configurations of the saliency-based defect detector when these are used with the datasets of the vessel compartments affected by corrosion. To be precise, Fig. 5.16 [top] provides the results obtained for the topside tank dataset, while Fig. 5.16 [middle] shows the curves corresponding to the forepeak tank dataset. Additionally, Fig. 5.16 [bottom] provides the metrics obtained when analysing the images from both ballast tanks.

Unlike what happened with the original corrosion dataset used in Chapter 4, symmetry outperforms contrast for the datasets comprising images taken by the MAV in the topside or/and the forepeak tanks. On the one hand, the ROC curve obtained using symmetry

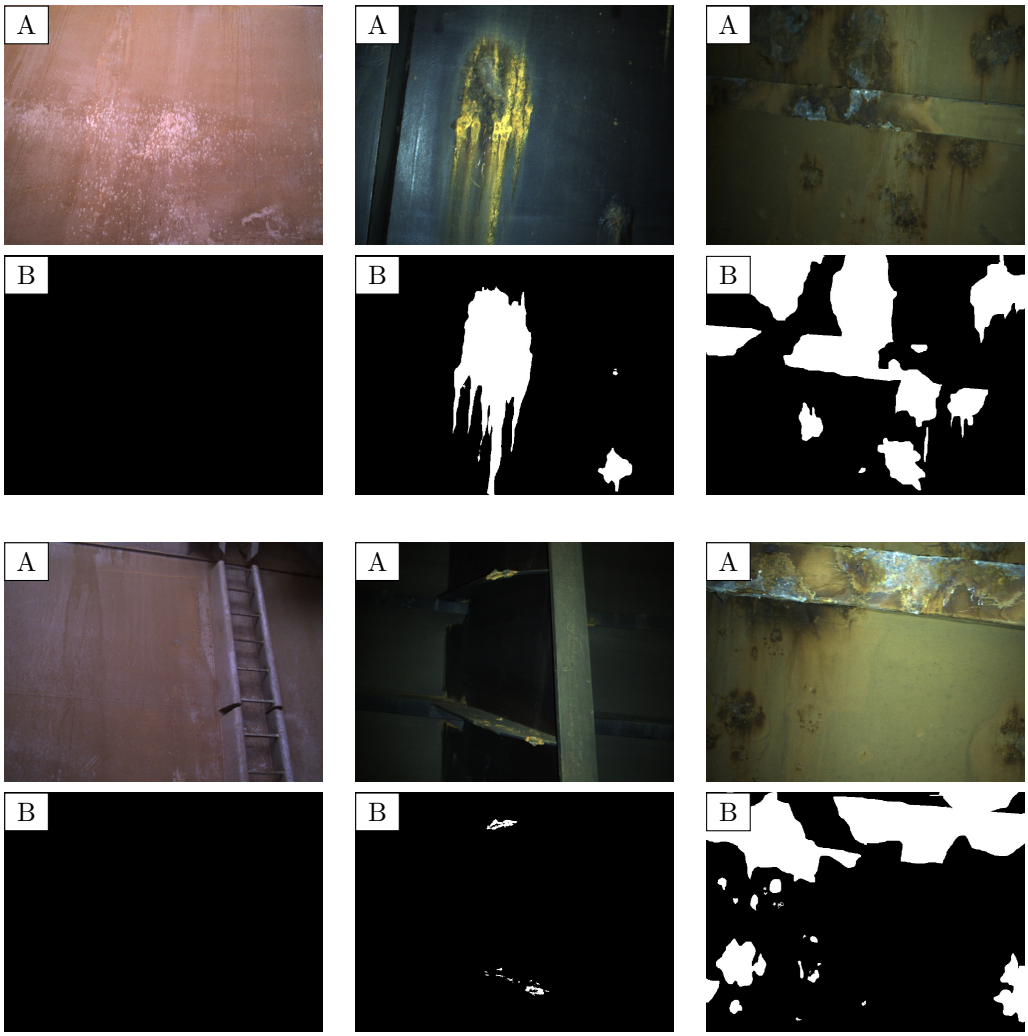


Figure 5.15: Some images taken by the MAV used to create the datasets: (A) pictures taken from the cargo hold (left), the topside tank (middle) and forepeak tank (right), (B) hand-labelled ground truth images.

goes closer to the (0, 1) corner than the curve obtained when using contrast. On the other hand, symmetry provides considerably higher values of precision. Regarding the combined methods, the OR and AND combinations provide very similar results, and outperform all the other single and combined versions of the detector. Nevertheless, the ORA combination presents slightly poorer performance due to the excessive importance given to the different channels of contrast: intensity, colour and orientation (see Section 4.5.3 for details). Table 5.1 provides the AUC values computed from the ROC curves shown in Fig. 5.16

A final experiment with the saliency-based general defect detector has been performed including also the images from the cargo hold dataset, so that all the images from the three vessel compartments have been considered. The performance metrics for this experiment can be found in Fig. 5.17. Remember that the images from the cargo hold do not present cor-

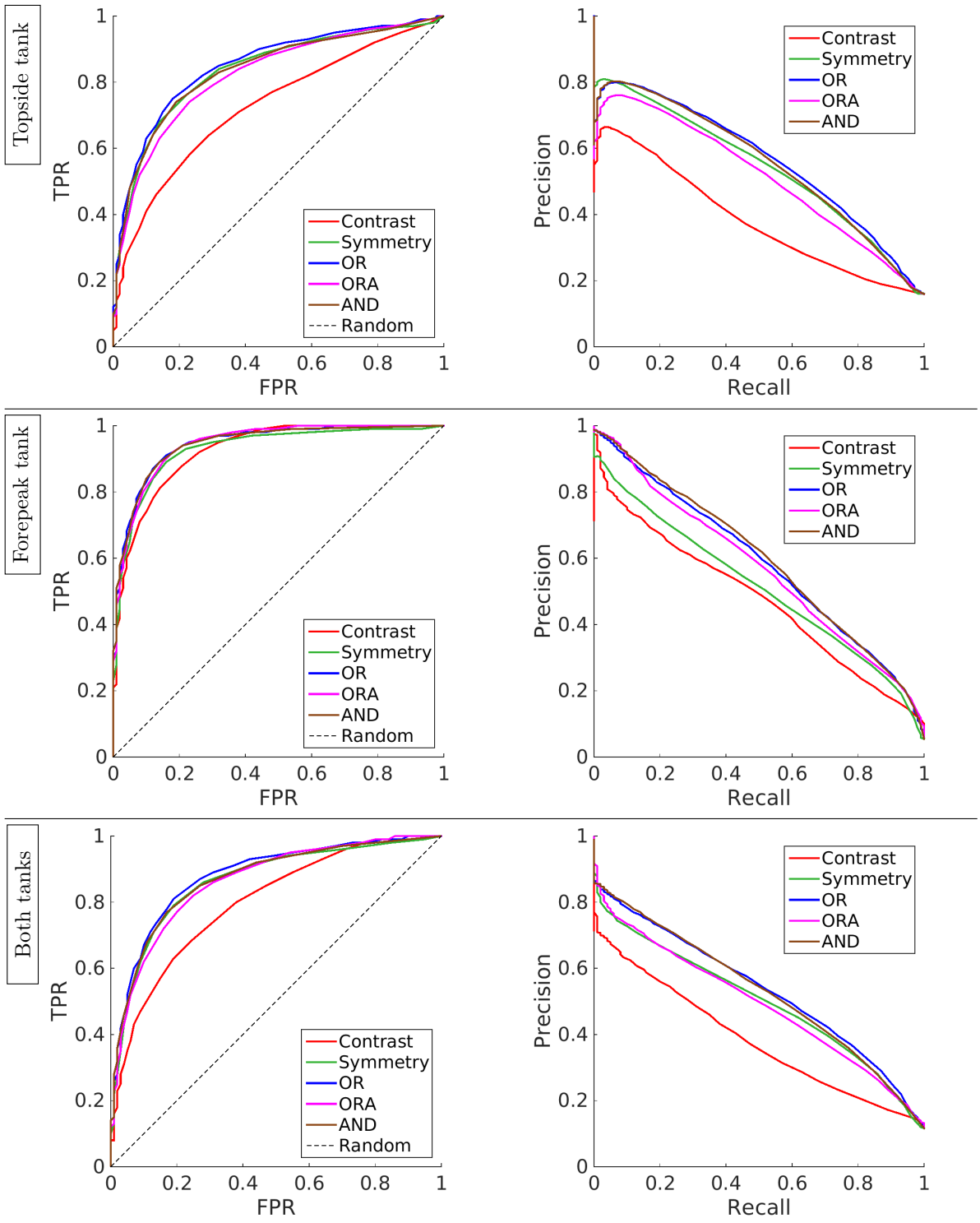


Figure 5.16: Performance of the saliency-based defect detector inspecting the images from the different vessel compartments affected by corrosion: (left) ROC curves and (right) PR curves. Each row corresponds to a different image dataset: (top) topside tank dataset, (middle) forepeak tank dataset, and (bottom) dataset resulting from merging the previous two datasets.

Table 5.1: AUC values for the saliency-based defect detector with the images taken using the MAV in the different vessel compartments. These values correspond to the ROC curves presented in Fig. 5.16 and Fig. 5.17.

	Contrast	**Symmetry**	**OR**	**ORA**	**AND**
Topside tank	0.723	0.836	0.848	0.820	0.836
Forepeak tank	0.839	0.930	0.931	0.925	0.941
Both tanks	0.760	0.863	0.873	0.854	0.869
Three compartments	0.776	0.816	0.852	0.840	0.843

Both tanks refers to the topside and the forepeak tanks.
Three compartments refers to the cargo hold, the topside tank and the forepeak tank.

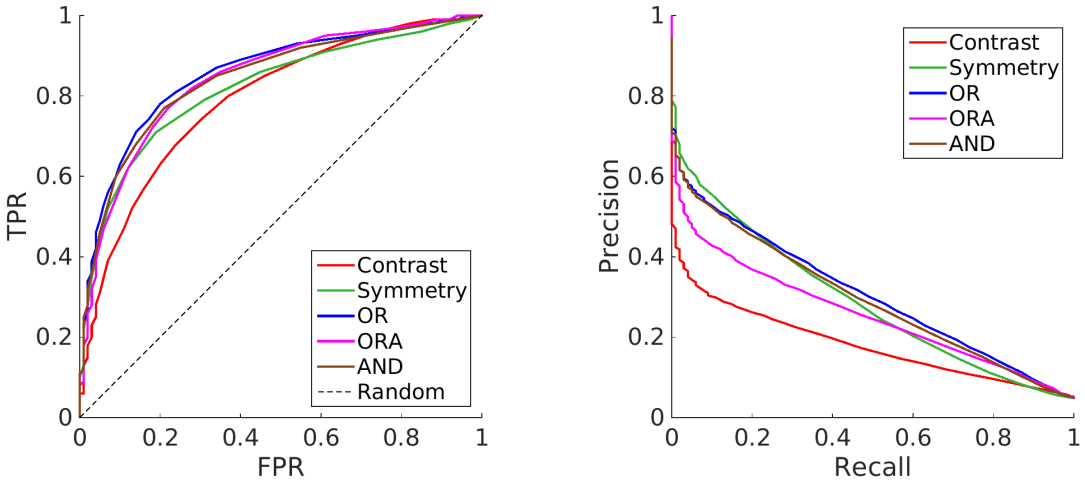


Figure 5.17: Performance of the saliency-based defect detector inspecting the images included in the three datasets (cargo hold, topside tank and forepeak tank): (left) ROC curves and (right) PR curves.

roded areas, so that any positive detection dramatically increases the FPR and decreases the precision. In the ROC space, the three combined methods again provide better performance than the single-feature detectors. Regarding the PR curve, the OR and AND combinations outperform all the other versions of the detector, while the precision of the ORA combination is more reduced due to the poorer performance provided by the contrast-based method. The AUC values computed for the resulting ROC curves are provided in the last row of Table 5.1.

The results obtained for the different versions of the saliency-based defect detector indicate that the performance of contrast and symmetry for detecting corroded areas in vessel structures vary depending on the compartment considered and the appearance of its rust. Nevertheless, the combination of both features provides good detection results in all the compartments.

The complete saliency-boosted corrosion detector has been used to inspect the images

Table 5.2: Performance metrics of the corrosion detector with images taken using the MAV in the different vessel compartments.

	TPR (recall)	**FPR**	**Precision**
Topside tank	0.758	0.159	0.460
Forepeak tank	0.876	0.073	0.394
Both tanks	0.781	0.122	0.444
Three compartments	0.780	0.050	0.431

Both tanks refers to the topside and the forepeak tanks.
Three compartments refers to the cargo hold, the topside tank and the forepeak tank.

from the three datasets. To do that, the detection method has been used as configured in Section 4.6.1, that is, using $\tau_E = 0.4$, $\tau_D = 0.175$ and the dictionary with 100 codewords previously created. Table 5.2 provides the metrics obtained for the different dataset combinations already used in Table 5.1. As can be observed, the corrosion detector yields a good performance for the images of the bulk carrier taken using the MAV. The TPR is above 75% in all cases, and reaches 78% when all the images from the three datasets are considered. The FPR for this case is 5%, and it is below 16% in the worst case, which corresponds to the topside tank dataset. Regarding precision, it is situated between 39% and 46% for the different dataset combinations.

By way of example, Fig 5.18 shows corrosion detection outputs for some of the images from the bulk carrier. As can be observed, the detection mostly coincides with the ground truth.

5.4 Conclusions

The field trials on board a bulk carrier allowed us to check the usability and performance of the technological tools developed to assist during vessel visual inspection. Regarding the robotic platform, it resulted very useful to take pictures of the highest parts of the vessel compartments without the need of using scaffoldings or cherry-pickers. Furthermore, its control architecture based on SA, allowed the user/pilot to focus on the inspection task, while the vehicle was in charge of its self-preservation. This is of particular interest when inspecting ballast tanks or other narrow spaces, where the vehicle is operated very close to the vessel structures.

The laser scanner-based positioning system, used to tag the images, provided good position estimations in general. It failed when the SLAM algorithm got confused due to the lack of distinguishable structures in the environment, such as corners or other irregular surfaces. Inside a bulk carrier, this was only observed in one of the experiments, which was performed inside the cargo hold, flying far from all its corners. Since the estimated position is not used by the control system, this failure is not critical and the safety of the platform is not

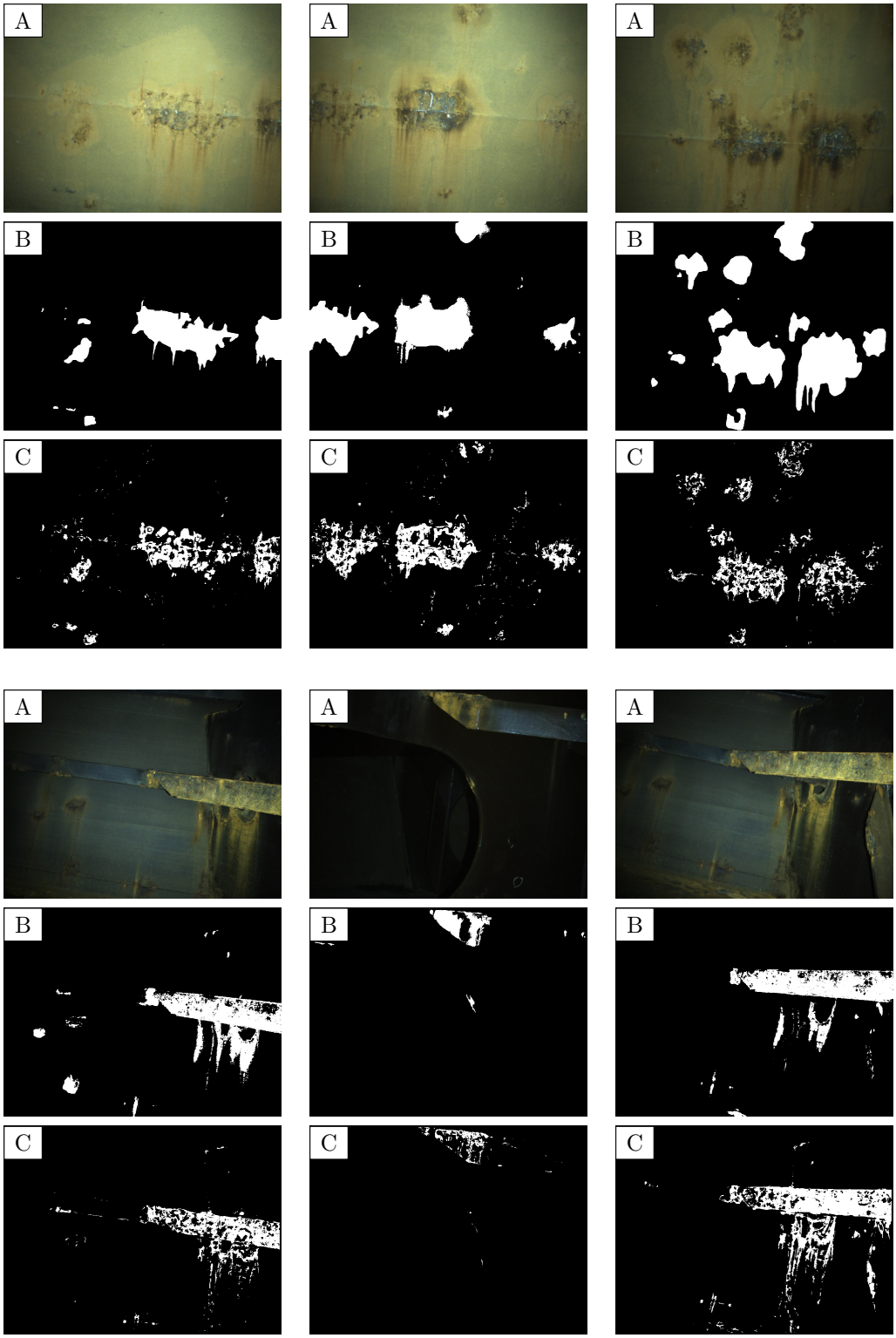


Figure 5.18: Corrosion detection results for some images from the bulk carrier: (A) input image, (B) ground truth, (C) corrosion detected.

compromised.

Regarding the corrosion detector, it has produced successful results for the images taken using the MAV. Unlike what happened with the original corrosion dataset used in Chapter 4, symmetry has provided better detection results than contrast for the pictures taken from the bulk carrier. This makes more interesting the combination of contrast and symmetry, and proves that these two features convey complementary information which can be combined to attain a higher detection performance in a wider range of situations.

As a final comment, the corrosion detector resulted in different performance levels for each of the compartments of the vessel. Among them, the highest TPR and lowest FPR were attained with the forepeak tank dataset, probably due to the high contrast between the yellowish corroded areas and the blueish non-corroded plates of this compartment (see Fig. 5.14 for some examples).

Conclusions

6.1 Summary of the Thesis

The inspection of vessels is nowadays carried out at a great cost. It entails the use of yard's facilities, where the vessel is usually situated in a dry dock, the cleaning and ventilation of all its cargo holds, ballast tanks, etc., and the installation of the scaffolding or other movable platforms which allow the surveyors to get a close-up view of the different structures of the vessel hull. Furthermore, since the inspection is usually performed at an elevated height, this entails a certain risk for the surveyors.

In this dissertation, we have presented novel technological tools to assist during the visual inspection of vessels, in order to contribute to a sort of reengineering process. On the one hand, a new aerial robotic platform has been proposed to allow the surveyor to perform the visual inspection of the vessel hull from a safe and comfortable position. The vehicle is equipped with cameras which permit teleporting the surveyor to the area under inspection.

According to the survey provided in Section 2.1, the only previous aerial platform intended for the inspection of ship hulls is the one we described in [20]. In comparison to this approach, the new platform allows the introduction of the surveyor in the position control loop, so that he/she can directly indicate the displacement commands. Furthermore, the control architecture of the new vehicle relies on the estimation of its velocity, and does not require a position estimate. This reduces the risk of accident due to the lack or miss-interpretation of sensors data, since velocities are typically easier to estimate.

The system architecture, which has been designed around the SA paradigm, assigns all the safety related issues to the robot, so that the user/surveyor can focus on the inspection at hand. This paradigm also defines the communication between the vehicle and the human, which is performed in an intuitive and qualitative way. All this results in a safe and easy-to-use tool.

For the velocity estimation, we have reviewed different sensing possibilities which can be installed on-board MAVs. For our particular problem, we have proposed three different sensor suites. The SS1 is based on the use of two optical flow sensors to estimate the vehicle speed regarding the floor and/or the inspected surface. The SS2 makes use of a laser scanner to perform the speed estimation by means of an ICP procedure. Finally, the SS3 combines the

sensors from SS1 and SS2 to provide a proper velocity estimation where the laser scanner fails due to, for example, the "canyoning" effect.

We proposed a three-layered control architecture where each layer is in charge of providing suitable commands to the next one. The low-level layer performs attitude and thrust control, and provides the commands to the motors. The mid-level layer consists of four PID controllers in charge of the speed (in the three axis) and height control. Finally, the high-level layer consists of several robot behaviours which are organized in a hybrid competitive-cooperative framework, and are in charge of accomplishing the user intention, ensuring the platform safety, increasing the autonomy level and checking the flight viability.

To tag the pictures with the position of the platform, two different SLAM algorithms have been integrated, depending on the sensor suite employed. When the SS1 is used, the ORB-SLAM method has been employed with the images provided by the camera. When the SS2 or SS3 are in use, the GMapping algorithm is employed with the laser scans provided by the laser scanner.

Regarding the system requirements enumerated in Section 3.1, the platform fulfils them all as indicated in the following:

1. The vehicle allows a close-up view of the inspected surface. It is based on a multirotor UAV with capabilities for hovering and VTOL, and it is equipped with a small digital camera and, optionally, with an additional video camera. The camera module allows the user/pilot to take pictures and image sequences on demand. Furthermore, the *inspection mode* allows to obtain good-quality pictures since this keeps constant the distance of the vehicle to the inspected wall while prevents fast movements which may cause blurring.

2. The vehicle obeys the user/surveyor commands. As part of the SA framework, the user/pilot can provide displacement commands by means of a joystick/gamepad. These commands are received by the control architecture which tries to accomplish them, as far as possible.

3. The vehicle allows reaching the highest structures of the vessel hull. The sensors comprising the different sensor suites provide measurements with regard to the surfaces/structures situated below, in front of and at both sides of the robotic platform. Therefore, when flying far from the floor, the vehicle state can be estimated as far as it is operated relatively close to other surfaces, what is also a requirement for a proper visual inspection.

4. The vehicle can be operated inside rather narrow spaces, such as ballast tanks, since it is based on an electrically-powered MAV (less than 2 Kg). Furthermore, the control software prevents collisions with the vessel structures.

5. The vehicle can be operated in dark areas, where daylight can not penetrate. When using the SS2, the state estimation is based on measurements provided by a laser scanner,

which does not requires light. Furthermore, successful pictures can be taken thanks to the use of a high power LED available in the platform.

6. The vehicle prevents collisions with any surrounding obstacle. As part of the SA paradigm, the behaviour-based control architecture is in charge of preventing collisions with the vessel structures or other obstacles, so that the vehicle can not collide even when the user/pilot provides commands to do so.

7. The robotic platform can be operated by a non-expert user, who maybe has never used a similar device. This is also thanks the SA paradigm, which comprises the concepts *instructive feedback*, *qualitative instructions* and *qualitative explanations*, among others.

8. The vehicle implements some autonomous behaviours to alleviate the inspection task to the user/surveyor. The *go-ahead* behaviour keeps the user speed command until some obstacle is detected in the proximity of the vehicle, or until the user provides a new speed command. This is of particular interest when a large displacement has to be performed such as, for example, when the vehicle has to be commanded to the other end of a big cargo hold where the next inspection is going to be performed. Similarly, the *inspection-ahead* behaviour allows keeping the user speed command while a large wall is being inspected using the *inspection mode*.

The different control systems and modules have been evaluated both in the laboratory and during field trials on board a real vessel. In the laboratory, three different implementations of the control architecture have been considered, using three MAVs with different payload capacities. We have evaluated their hovering and displacement capabilities, the performance of the different robot behaviours, their usability during an inspection task and the position estimation accuracy using the two SLAM methods considered. Successful results have been obtained for all the laboratory experiments using any of the three MAVs.

On board the vessel, the platform equipped with the SS2 has been used since this can be also operated in poorly illuminated compartments. The behaviour of the platform during the field test was similar to what was observed during the laboratory experiments. Nevertheless, the SLAM method got confused during one of the experiments inside the cargo hold, probably due to the large distance to all its corners (see Fig. 5.7). Nevertheless, since the estimated position is just used to tag the images and it is not required by the control architecture, the estimation errors are not critical and do not compromise the platform safety.

On the other hand, several defect detectors specifically devised for vessel visual inspection have been proposed. In first place, different approaches for corrosion detection have been presented and evaluated. They are based on the combination of different features to describe the colour and texture of corroded surfaces. The different methods have been evaluated with the same dataset and all of them have produced relatively good results. The selection of one method or another depends on whether we prefer to obtain the best classification ratios or whether we require a very fast answer.

In second place, we have proposed methods for crack detection on vessel structures. They combine an edge detection method with a region-growing procedure to detect cracks in a two-step process. In the first step, potential cracks or crack portions are identified as elongated collections of pixels which are darker than their neighbours. In the second step, the connected potential cracks are combined into larger units which are finally labelled as cracks if they are still elongated. Unlike other methods, our approach does not require a fixed distance to the inspected surface since the crack is not supposed to be of a specific size. Similarly, in our approach, the "darkness" of the crack is evaluated with regard to its surrounding area, instead of using a fixed threshold in the gray-scale space. The proposed method is able to detect all the cracks from the dataset; however it also leads to some false positives mostly produced by shadows in the image. To reduce these classification errors, we propose to guide the crack inspection using the output provided by a corrosion detector, since most cracks of the dataset take place over corroded areas. The results obtained with the corrosion-guided version considerably reduce the false positive detections. Nevertheless, this version is not able to detect a few cracks from the dataset which do not lie over corrosion.

In third place, we have evaluated the use of saliency methods for detecting generic defects on vessel structures. Contrast and symmetry features have been combined within two different frameworks for that purpose. On the one hand, a generic framework has been proposed to combine, in a flexible way, these features, in order to assess different combination operators. On the other hand, a Bayesian framework has been applied to combine, in a probabilistic way, the information previously obtained about these two features. Different configurations of both approaches have been evaluated with the same dataset, and good performance results have been obtained. In all cases, the configurations which combine both features have provided the best results. The classification performance obtained with both frameworks is very similar, so that the generic framework is considered better since it does not require a previous learning stage.

Finally, one of the saliency-based generic defect detectors has been combined with a corrosion and a crack detectors in order to boost their detection performances. The results show that the saliency-boosted versions effectively reduce their false positive detections and, thus, increase their precision.

The boosted version of the corrosion detector has been also used during the field trials on board a real vessel, to inspect the images taken using the robotic platform. The results obtained for the different vessel compartments allow to confirm that contrast and symmetry provide complementary information, and that their combination allows achieving a higher detection performance in a wider range of situations.

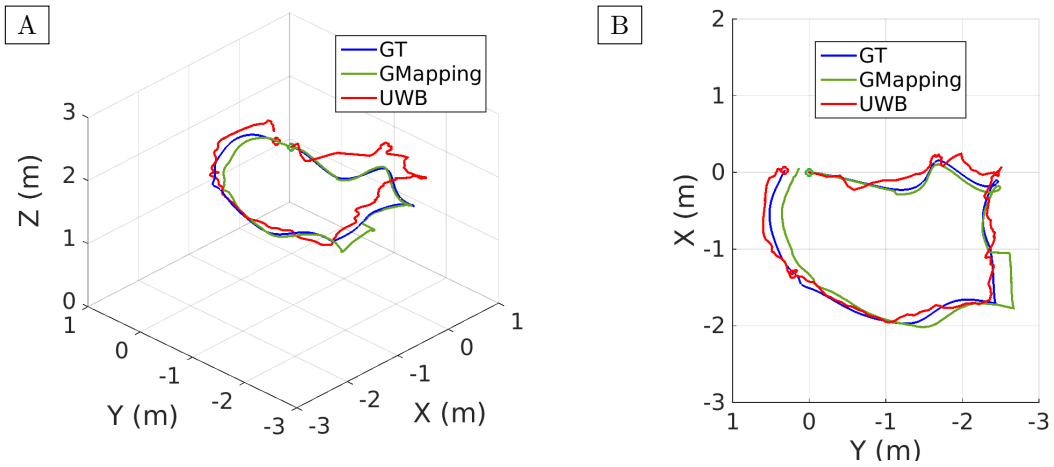


Figure 6.1: Estimated path provided by a UWB system during a flight with the MAV. It is compared with the trajectory provided by the GMapping method, and the ground truth provided by the motion capture system: (A) 3D plot of the trajectory, (B) 2D projection of the trajectory. The green and red dots indicate the initial and final points respectively.

6.2 Future Work

The following tasks are planned to be carried out as future work:

- In order to prevent miss-estimations of the vehicle velocity/position when using the SS2, we plan to implement an additional robot behaviour in charge of keeping the platform close to at least two distinguishable walls or structures. The idea is to ensure that the laser-based odometer and SLAM methods are able to provide a proper estimate of the displacement of the vehicle.

- To overcome the errors in the position estimated by the SLAM methods, we also plan to merge their estimates with the position provided by a global positioning system based on *Ultra-Wide Band* (UWB). Some preliminary tests have been already performed with an UWB system comprising four anchors and one tag, which is installed on board the MAV. In this regard, Fig. 6.1 compares the positions provided by the GMapping method, the UWB system, and the motion capture system (which is considered as the ground truth). Despite the position provided by the UWB system has been filtered using a mean filter, it still presents some noise. Nevertheless, it is worth trying to merge these results, and the measures provided by other sensors (e.g. IMU), using a KF.

- We also plan to implement other behaviours to increase the platform autonomy. For example, we consider some behaviours to assure the complete coverage of the surface under inspection. This would be useful, for example, to create a reconstruction of the surface.

- As a last point related with aerial platform, we want to formally verify the control software.
- Regarding defect inspection, we plan to quantify the defective area in the inspected surface, what probably entails its 3D reconstruction (at least partially).
- We also plan to study the possibility to identify the kind and position of the structural element under inspection: stiffener, webframe, bulkhead, etc.
- Finally, we plan to apply deep-learning techniques for corrosion and crack detection in vessel structures.
- Apart from the research targets, we plan to make the aerial platform more robust to crashes, more water resistant and, maybe, explosion-proof.

Bibliography

[1] United Nations Conference on Trade and Development, *Review of Maritime Transport*. United Nations Publication, 2015, UNCTAD/RMT/2015.

[2] INCASS Consortium, "D2.1: Technological Tools and Components for the Conduction of Automated or Supported Survey Activities," 2015, EU FP7 Project. GA 605200.

[3] A. Ortiz, F. Bonnin-Pascual, E. Garcia-Fidalgo, and J. P. Company-Corcoles, "Vision-Based Corrosion Detection Assisted by a Micro-Aerial Vehicle in a Vessel Inspection Application," *Sensors*, vol. 16, no. 2118, 2016.

[4] F. Bonnin-Pascual and A. Ortiz, "A Flying Tool for Sensing Vessel Structure Defects using Image Contrast-based Saliency," *IEEE Sensors Journal*, vol. 16, no. 15, pp. 6114–6121, 2016.

[5] F. Bonnin-Pascual and A. Ortiz, "A Saliency-boosted Corrosion Detector for the Visual Inspection of Vessels," in *Artificial Intelligence Research and Development*. IOS Press, 2017, (in press).

[6] A. Ortiz, F. Bonnin-Pascual, E. Garcia-Fidalgo, and J. P. Company-Corcoles, "Defect-level Inspection Aids for Automated Vessel Visual Inspection," in *Jornadas Automar (Marine Automation Workshop)*, 2017.

[7] A. Ortiz, F. Bonnin-Pascual, E. Garcia-Fidalgo, and J. P. Company-Corcoles, "The INCASS Project Approach towards Automated Visual Inspection of Vessels," in *Jornadas Nacionales de Robótica (Spanish Robotics Workshop)*, 2017.

[8] A. Ortiz, F. Bonnin-Pascual, E. Garcia-Fidalgo, and J. P. Company-Corcoles, "Towards Automated Ship Inspection: A Visual Data-Oriented Toolbox," in *International Conference on Maritime Safety and Operations*, 2016.

[9] F. Bonnin-Pascual and A. Ortiz, "A Generic Framework for Defect Detection on Vessel Structures based on Image Saliency," in *IEEE International Conference on Emerging Technologies and Factory Automation*, 2016.

[10] T. Koch, S. Natarajan, F. Bernhard, A. Ortiz, F. Bonnin-Pascual, E. Garcia-Fidalgo, and J. P. Company-Corcoles, "Advances in Automated Ship Structure Inspection," in *International Conference on Computer Applications and Information Technology in the Maritime Industries*, 2016.

[11] A. Ortiz, F. Bonnin-Pascual, E. Garcia-Fidalgo, and J. P. Company-Corcoles, "Visual Inspection of Vessels by means of a Micro-Aerial Vehicle: an Artificial Neural Network Approach for Corrosion Detection," in *Robot 2015: Second Iberian Robotics Conference. Advances in Robotics*, L. P. Reis, A. P. Moreira, P. U. Lima, L. Montano, and V. Muñoz-Martinez, Eds. Springer International Publishing, 2015, vol. 418, pp. 223–234.

[12] A. Ortiz, F. Bonnin-Pascual, E. Garcia-Fidalgo, and J. P. Company-Corcoles, "Saliency-driven Visual Inspection of Vessels by means of a Multirotor," in *Workshop on Vision-based Control and Navigation of Small, Lightweight UAVs (IROS)*, 2015.

[13] F. Bonnin-Pascual, A. Ortiz, E. Garcia-Fidalgo, and J. P. Company, "A Micro-Aerial Platform for Vessel Visual Inspection based on Supervised Autonomy," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 46–52.

[14] F. Bonnin-Pascual and A. Ortiz, "A Probabilistic Approach for Defect Detection based on Saliency Mechanisms," in *IEEE International Conference on Emerging Technologies and Factory Automation*, 2014.

[15] F. Bonnin-Pascual and A. Ortiz, "Detection of Defects on Vessel Structures using Saliency-related Features," Department of Mathematics and Computer Science, University of the Balearic Islands, Tech. Rep. A-04-2015, 2015. [Online]. Available: http://dmi.uib.es/~xbonnin/static/papers/techrepA042015_Bonnin2015.pdf

[16] F. Bonnin-Pascual, A. Ortiz, E. Garcia-Fidalgo, and J. P. Company, "A Micro-Aerial Vehicle based on Supervised Autonomy for Vessel Visual Inspection," Department of Mathematics and Computer Science, University of the Balearic Islands, Tech. Rep. A-02-2015, 2015. [Online]. Available: http://dmi.uib.es/~xbonnin/static/papers/techrepA022015_Bonnin2015.pdf

[17] M. Eich, F. Bonnin-Pascual, E. Garcia-Fidalgo, A. Ortiz, G. Bruzzone, Y. Koveos, and F. Kirchner, "A Robot Application to Marine Vessel Inspection," *Journal of Field Robotics*, vol. 31, no. 2, pp. 319–341, 2014.

[18] A. Ortiz, F. Bonnin-Pascual, and E. Garcia-Fidalgo, "Vessel Inspection: A Micro-Aerial Vehicle-based Approach," *Journal of Intelligent and Robotic Systems*, vol. 76, pp. 151–167, 2014.

[19] F. Bonnin-Pascual and A. Ortiz, "Corrosion Detection for Automated Visual Inspection," in *Developments in Corrosion Protection*, D. M. Aliofkhazraei, Ed. InTech, 2014, ch. 25, pp. 619–632.

[20] F. Bonnin-Pascual, E. Garcia-Fidalgo, and A. Ortiz, "Semi-autonomous Visual Inspection of Vessels Assisted by an Unmanned Micro Aerial Vehicle," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 3955–3961.

[21] E. Garcia-Fidalgo, F. Bonnin-Pascual, and A. Ortiz, "A Control Architecture for a Micro Aerial Vehicle Intended for Vessel Visual Inspection," in *Jornadas de Computación Empotrada*, 2012.

[22] A. Ortiz, E. Garcia-Fidalgo, and F. Bonnin-Pascual, "A Micro Aerial Vehicle for Vessel Visual Inspection Assistance," in *International Conference on Computer Applications and Information Technology in the Maritime Industries*, 2012.

[23] A. Ortiz, F. Bonnin-Pascual, and E. Garcia-Fidalgo, "On the Use of UAVs for Vessel Inspection Assistance," in *Workshop on Research, Development and Education on Unmanned Aerial Systems*, 2011.

[24] F. Bonnin-Pascual and A. Ortiz, "An AdaBoost-based Approach for Coating Breakdown Detection in Metallic Surfaces," in *IEEE Mediterranean Conference on Control and Automation*, 2011.

[25] F. Bonnin-Pascual and A. Ortiz, "Combination of Weak Classifiers for Metallic Corrosion Detection and Guided Crack Location," in *IEEE International Conference on Emerging Technologies and Factory Automation*, 2010.

[26] A. Ortiz, F. Bonnin-Pascual, A. Gibbins, P. Apostolopoulou, W. Bateman, M. Eich, F. Spadoni, M. Caccia, and L. Drikos, "First Steps Towards a Roboticized Visual Inspection System for Vessels," in *IEEE International Conference on Emerging Technologies and Factory Automation*, 2010, pp. 1–6.

[27] F. Bonnin-Pascual and A. Ortiz, "Detection of Cracks and Corrosion for Automated Vessels Visual Inspection," in *Artificial Intelligence Research and Development*, R. Alquezar, A. Moreno, and J. Aguilar, Eds. IOS Press, 2010, pp. 111–120.

[28] A. Ortiz, F. Bonnin-Pascual, and E. Garcia-Fidalgo, "Vessel Inspection Assistance by means of a Micro-Aerial Vehicle: Control Architecture and Self-Localization Issues," Department of Mathematics and Computer Science, University of the Balearic Islands, Tech. Rep. A-02-2013, 2013. [Online]. Available: http://dmi.uib.es/~xbonnin/static/papers/techrepA022013_Bonnin2013.pdf

[29] E. Garcia-Fidalgo, A. Ortiz, F. Bonnin-Pascual, and J. P. Company, "Fast Image Mosaicing using Incremental Bags of Binary Words," in *IEEE International Conference on Robotics and Automation*, 2016.

[30] E. Garcia-Fidalgo, A. Ortiz, F. Bonnin-Pascual, and J. P. Company, "A Mosaicing Approach for Vessel Visual Inspection using a Micro-Aerial Vehicle," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 104–110.

[31] A. Ortiz, F. Bonnin-Pascual, E. Garcia-Fidalgo, and J. P. Beltran, "A Control Software Architecture for Autonomous Unmanned Vehicles inspired in Generic Components," in *IEEE Mediterranean Conference on Control and Automation*, 2011.

[32] E. Garcia-Fidalgo, A. Ortiz, F. Bonnin-Pascual, and J. P. Company, "A Multi-Threaded Architecture for Fast Topology Estimation in Image Mosaicing," Department of Mathematics and Computer Science, University of the Balearic Islands, Palma de Mallorca, Tech. Rep. A-05-2015, 2015. [Online]. Available: http://dmi.uib.es/~egarcia/static/papers/techrepA052015_Garcia2015.pdf

[33] E. Garcia-Fidalgo, A. Ortiz, F. Bonnin-Pascual, and J. P. Company, "Vessel Visual Inspection: A Mosaicing Approach," Department of Mathematics and Computer Science, University of the Balearic Islands, Palma de Mallorca, Tech. Rep. A-01-2015, March 2015. [Online]. Available: http://dmi.uib.es/~egarcia/static/papers/techrepA012015_Garcia2015.pdf

[34] J. Katrašnik, F. Pernuš, and B. Likar, "A Survey of Mobile Robots for Distribution Power Line Inspection," *IEEE Transactions on Power Delivery*, vol. 25, no. 1, pp. 485–493, 2010.

[35] A. Pagnano, M. Höpf, and R. Teti, "A Roadmap for Automated Power Line Inspection. Maintenance and Repair," in *CIRP Conference on Intelligent Computation in Manufacturing Engineering*, 2013, pp. 234–239.

[36] P. Ridao, M. Carreras, D. Ribas, and R. Garcia, "Visual Inspection of Hydroelectric Dams using an Autonomous Underwater Vehicle," *Journal of Field Robotics*, vol. 27, no. 6, pp. 759–778, 2010.

[37] N. A. Cruz, A. C. Matos, R. M. Almeida, B. M. Ferreira, and N. Abreu, "TriMARES - a Hybrid AUV/ROV for Dam Inspection," in *IEEE/MTS OCEANS Conference*, 2011, pp. 1–7.

[38] H. M. La, R. S. Lim, B. Basily, N. Gucunski, J. Yi, A. Maher, F. A. Romero, and H. Parvardeh, "Autonomous Robotic System for High-Efficiency Non-Destructive Bridge Deck Inspection and Evaluation," in *IEEE International Conference on Automation Science and Engineering*, 2013, pp. 1053–1058.

[39] R. S. Lim, H. M. La, and W. Sheng, "A Robotic Crack Inspection and Mapping System for Bridge Deck Maintenance," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 2, pp. 367–378, 2014.

[40] J. M. Mirats and W. Garthwaite, "Robotic Devices for Water Main In-Pipe Inspection: A Survey," *Journal of Field Robotics*, vol. 27, no. 4, pp. 491–508, 2010.

[41] N. S. Roslin, A. Anuar, M. F. A. Jalal, and K. S. M. Sahari, "A Review: Hybrid Locomotion of In-pipe Inspection Robot," in *International Symposium on Robotics and Intelligent Sensors*, vol. 41, 2012, pp. 1456–1462.

[42] M. Siegel and P. Gunatilake, "Remote Enhanced Visual Inspection of Aircraft by a Mobile Robot," in *IEEE Workshop on Emerging Technologies, Intelligent Measurement and Virtual Systems for Instrumentation and Measurement*, 1998.

[43] T. S. White, R. Alexander, G. Callow, A. Cooke, S. Harris, and J. Sargent, "A Mobile Climbing Robot for High Precision Manufacture and Inspection of Aerostructures," *International Journal of Robotics Research*, vol. 24, no. 7, pp. 589–598, 2005.

[44] D. C. Lynn and G. S. Bohlander, "Performing Ship Hull Inspections using a Remotely Operated Vehicle," in *IEEE/MTS OCEANS Conference*, vol. 2, 1999, pp. 555–562.

[45] S. M. Newsome and J. Rodocker, "Effective Technology for Underwater Hull and Infrastructure Inspection," in *IEEE/MTS OCEANS Conference*, 2009, pp. 1–6.

[46] K. Ishizu, N. Sakagami, K. Ishimaru, M. Shibata, H. Onishi, S. Murakami, and S. Kawamura, "Ship Hull Inspection using A Small Underwater Robot With A Mechanical Contact Mechanism," in *IEEE/MTS OCEANS Conference*, 2012, pp. 1–6.

[47] S. E. Harris and E. V. Slate, "Lamp Ray: Ship Hull Assessment for Value, Safety and Readiness," in *IEEE/MTS OCEANS Conference*, 1999, pp. 493–500.

[48] E. D'Amaddio, S. Harris, and E. Bergeron, E. amd Slate, "Method and apparatus for inspecting a submerged structure," Patent U.S. Patent 6,317,387, 2001.

[49] T. S. Akinfiev, M. A. Armada, and R. Fernandez, "Nondestructive Testing of the State of a Ship's Hull with an Underwater Robot," *Russian Journal of Nondestructive Testing*, vol. 44, no. 9, pp. 626–633, 2008.

[50] M. Narewski, "Hismar - Underwater Hull Inspection and Cleaning System As a Tool for Ship Propulsion System Performance Increase," *Journal of Polish CIMAC*, vol. 4, no. 2, pp. 227–234, 2009.

[51] C. Z. Ferreira, G. Y. C. Conte, J. P. J. Avila, R. C. Pereira, and T. M. C. Ribeiro, "Underwater Robotic Vehicle for Ship Hull Inspection: Control System Architecture," in *International Congress of Mechanical Engineering*, 2013.

[52] G. M. Trimble and E. O. Belcher, "Ship Berthing and Hull Inspection using the CetusII AUV and MIRIS High-Resolution Sonar," in *IEEE/MTS OCEANS Conference*, 2002.

[53] J. Vaganay, M. L. Elkins, S. Willcox, F. S. Hover, R. S. Damus, S. Desset, J. P. Morash, and V. C. Polidoro, "Ship Hull Inspection by Hull-Relative Navigation and Control," in *IEEE/MTS OCEANS Conference*, 2005.

[54] J. Vaganay, M. Elkins, D. Esposito, W. O'Halloran, F. Hover, and M. Kokko, "Ship Hull Inspection with the HAUV: US Navy and NATO Demonstrations Results," in *IEEE/MTS OCEANS Conference*, 2006.

[55] F. Hover, J. Vaganay, M. Elkins, S. Willcox, V. Polidoro, J. Morash, R. Damus, and S. Desset, "A Vehicle System for Autonomous Relative Survey of In-Water Ships," *Marine Technology Society Journal*, vol. 41, no. 2, pp. 44–55, 2007.

[56] Y. Li, Y. Pang, Y. Chen, L. Wan, and J. Zou, "A Hull-Inspect ROV Control System Architecture," *China Ocean Engineering*, vol. 23, no. 4, pp. 751–761, 2009.

[57] G. E. Packard, R. Stokey, R. Christenson, F. Jaffre, M. Purcell, and R. Littlefield, "Hull Inspection and Confined Area Search Capabilities of REMUS Autonomous Underwater Vehicle," in *IEEE/MTS OCEANS Conference*, 2010.

[58] E. Belcher, W. Hanot, and J. Burch, "Dual-Frequency Identification Sonar (DIDSON)," in *International Symposium on Underwater Technology*, 2002, pp. 187–192.

[59] S. Reed, A. Cormack, K. Hamilton, I. Tena Ruiz, and D. Lane, "Automatic Ship Hull Inspection using Unmanned Underwater Vehicles (UUV's)," in *International Symposium on Technology and the Mine Problem*, 2006.

[60] M. A. Kokko, "Range-based Navigation of AUVs Operating Near Ship Hulls," Master's thesis, Massachusetts Institute of Technology, 2007. [Online]. Available: https://dspace.mit.edu/handle/1721.1/40292

[61] M. Walter, F. Hover, and J. Leonard, "SLAM for Ship Hull Inspection using Exactly Sparse Extended Information Filters," in *IEEE International Conference on Robotics and Automation*, 2008, pp. 1463–1470.

[62] H. Durrant-Whyte and T. Bailey, "Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms," *IEEE Robotics and Automation Magazine*, vol. 2, pp. 99–110, 2006.

[63] M. VanMiddlesworth, M. Kaess, F. Hover, and J. J. Leonard, "Mapping 3D Underwater Environments with Smoothed Submaps," in *International Conference on Field and Service Robotics*, 2013, pp. 17–30.

[64] Z. Zainal Abidin and M. R. Arshad, "Visual Servoing with Application to ROV for Ship Hull Inspection," in *International Conference on Man-Machine Systems*, 2006.

[65] S. Negahdaripour and P. Firoozfam, "An ROV Stereovision System for Ship-Hull Inspection," *IEEE Journal of Oceanic Engineering*, vol. 31, no. 3, pp. 551–564, 2006.

[66] R. Schattschneider, G. Maurino, and W. Wang, "Towards Stereo Vision SLAM based Pose Estimation for Ship Hull Inspection," in *IEEE/MTS OCEANS Conference*, 2011.

[67] F. S. Hover, R. M. Eustice, A. Kim, B. Englot, H. Johannsson, M. Kaess, and J. J. Leonard, "Advanced Perception, Navigation and Planning for Autonomous In-Water Ship Hull Inspection," *International Journal of Robotics Research*, vol. 31, no. 12, pp. 1445–1464, 2012.

[68] H. Johannsson, M. Kaess, B. Englot, F. Hover, and J. Leonard, "Imaging Sonar-aided Navigation for Autonomous Underwater Harbor Surveillance," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 4396–4403.

[69] A. Kim and R. Eustice, "Pose-graph Visual SLAM with Geometric Model Selection for Autonomous Underwater Ship Hull Inspection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 1559–1565.

[70] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental Smoothing and Mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[71] A. Kim and R. M. Eustice, "Real-Time Visual SLAM for Autonomous Underwater Hull Inspection using Visual Saliency," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 719–733, 2013.

[72] P. Ozog, N. Carlevaris-Bianco, A. Kim, and R. M. Eustice, "Long-term Mapping Techniques for Ship Hull Inspection and Surveillance using an Autonomous Underwater Vehicle," *Journal of Field Robotics*, vol. 33, no. 3, pp. 265–289, 2016.

[73] P. Ozog and R. M. Eustice, "Identifying Structural Anomalies in Image Reconstructions of Underwater Ship Hulls," in *IEEE/MTS OCEANS Conference*, 2015.

[74] L. L. Menegaldo, M. Santos, G. A. N. Ferreira, R. G. Siqueira, and L. Moscato, "SIRUS: A Mobile Robot for Floating Production Storage and Offloading (FPSO) Ship Hull Inspection," in *IEEE International Workshop on Advanced Motion Control*, 2008, pp. 27–32.

[75] M. Bibuli, G. Bruzzone, G. Bruzzone, M. Caccia, M. Giacopelli, A. Petitti, and E. Spirandelli, "MARC: Magnetic Autonomous Robotic Crawler Development and Exploitation in the MINOAS Project," in *International Conference on Computer Applications and Information Technology in the Maritime Industries*, 2012, pp. 62–75.

[76] M. Eich and T. Vögele, "Design and Control of a Lightweight Magnetic Climbing Robot for Vessel Inspection," in *IEEE Mediterranean Conference on Control and Automation*, 2011, pp. 1200–1205.

[77] K. Fondahl, M. Eich, J. Wollenberg, and F. Kirchner, "A Magnetic Climbing Robot for Marine Inspection Services," in *International Conference on Computer Applications and Information Technology in the Maritime Industries*, 2012, pp. 92–102.

[78] M. Ahmed, M. Eich, and F. Bernhard, "Design and Control of MIRA: a Lightweight Climbing Robot for Ship Inspection," *International Letters of Chemistry, Physics and Astronomy*, vol. 55, pp. 128–135, 2015.

[79] S. A. Nicinski, "Development of a Remotely Operated Ship Hull Inspection Vehicle," in *IEEE/MTS OCEANS Conference*, 1983, pp. 583–587.

[80] A. A. Carvalho, L. V. S. Sagrilo, I. C. Silva, J. M. A. Rebello, and R. O. Carneval, "On the Reliability of an Automated Ultrasonic System for Hull Inspection in Ship-based Oil Production Units," *Applied Ocean Research*, vol. 25, pp. 235–241, 2003.

[81] C. Huerzeler, G. Caprari, E. Zwicker, and L. Marconi, "Applying Aerial Robotics for Inspections of Power and Petrochemical Facilities," in *International Conference on Applied Robotics for the Power Industry*, 2012, pp. 167–172.

[82] G. Morgenthal and N. Hallermann, "Quality Assessment of Unmanned Aerial Vehicle (UAV) Based Visual Inspection of Structures," *Advances in Structural Engineering*, vol. 17, no. 3, pp. 289–302, 2014.

[83] C. Sampedro, C. Martinez, A. Chauhan, and P. Campoy, "A Supervised Approach to Electric Tower Detection and Classification for Power Line Inspection," in *IEEE World Congress on Computational Intelligence*, 2014.

[84] C. Martinez, C. Sampedro, A. Chauhan, and P. Campoy, "Towards Autonomous Detection and Tracking of Electric Towers for Aerial Power Line Inspection," in *International Conference on Unmanned Aircraft Systems*, 2014, pp. 284–295.

[85] N. S. Roberts, "Corrosion Detection in Enclosed Environments using Remote Systems," Master's thesis, Alfred University, 2016. [Online]. Available: http://hdl.handle.net/10829/7248

[86] P. B. Quater, F. Grimaccia, S. Leva, M. Mussetta, and M. Aghaei, "Light Unmanned Aerial Vehicles (UAVs) for Cooperative Inspection of PV Plants," *IEEE Journal of Photovoltaics*, vol. 4, no. 4, pp. 1107–1113, 2014.

[87] N. Hallermann and G. Morgenthal, "Visual Inspection Strategies for Large Bridges using Unmanned Aerial Vehicles (UAV)," in *International Conference on Bridge Maintenance, Safety and Management*, 2014.

[88] A. Ellenberg, A. Kontsos, F. Moon, and I. Bartoli, "Bridge Related Damage Quantification using Unmanned Aerial Vehicle Imagery," *Structural Control and Health Monitoring*, vol. 23, pp. 1168–1179, 2016.

[89] C. Eschmann, C. M. Kuo, C. H. Kuo, and C. Boller, "Unmanned Aircraft Systems for Remote Building Inspection and Monitoring," in *European Workshop on Structural Health Monitoring*, 2012, pp. 1–8.

[90] S.-s. Choi and E.-k. Kim, "Building Crack Inspection using Small UAV," in *International Conference on Advanced Communication Technology*, 2015, pp. 235–238.

[91] L. V. Campo, J. C. Corrales, and A. Ledezma, "An Aerial Autonomous Robot for Complete Coverage Outdoors," in *Workshop of Physical Agents*, 2016.

[92] N. E. Serrano, "Autonomous Quadrotor Unmanned Aerial Vehicle for Culvert Inspection," Master's thesis, Massachusetts Institute of Technology, 2011. [Online]. Available: http://hdl.handle.net/1721.1/67752

[93] N. Michael, S. Shen, K. Motha, Y. Mulgaonkar, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida, K. Ohno, E. Takeuchi, and S. Tadokoro, "Collaborative Mapping of an Earthquake-Damaged Building via Ground and Aerial Robots," *Journal of Field Robotics*, vol. 29, no. 5, pp. 832–841, 2012.

[94] M. Satler, M. Unetti, N. Giordani, C. A. Avizzano, and P. Tripicchio, "Towards an Autonomous Flying Robot for Inspections in Open and Constrained Spaces," in *Multi-Conference on Systems, Signals and Devices*, 2014.

[95] O. McAree, J. M. Aitken, and S. M. Veres, "A Model based Design Framework for Safety Verification of a Semi-Autonomous Inspection Drone," in *UK Automatic Control Conference*, 2016.

[96] M. Burri, J. Nikolic, C. Hürzeler, G. Caprari, and R. Siegwart, "Aerial Service Robots for Visual Inspection of Thermal Power Plant Boiler Systems," in *International Conference on Applied Robotics for the Power Industry*, 2012, pp. 70–75.

[97] J. Nikolic, M. Burri, J. Rehder, S. Leutenegger, C. Huerzeler, and R. Siegwart, "A UAV System for Inspection of Industrial Facilities," in *IEEE Aerospace Conference*, 2013, pp. 1–8.

[98] S. Omari, P. Gohl, M. Burri, M. Achtelik, and R. Siegwart, "Visual Industrial Inspection using Aerial Robots," in *International Conference on Applied Robotics for the Power Industry*, 2014, pp. 1–5.

[99] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart, "A Synchronized Visual-Inertial Sensor System with FPGA Pre-Processing for Accurate Real-Time SLAM," in *IEEE International Conference on Robotics and Automation*, 2014.

[100] P. Gohl, M. Burri, S. Omari, J. Rehder, J. Nikolic, M. Achtelik, and R. Siegwart, "Towards Autonomous Mine Inspection," in *International Conference on Applied Robotics for the Power Industry*, 2014.

[101] I. Sa, S. Hrabar, and P. Corke, "Inspection of Pole-Like Structures using a Visual-Inertial Aided VTOL Platform with Shared Autonomy," *Sensors*, vol. 15, no. 9, pp. 22 003–22 048, 2015.

[102] V. Lippiello and B. Siciliano, "Wall Inspection Control of a VTOL Unmanned Aerial Vehicle based on a Stereo Optical Flow," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 4296–4302.

[103] S. Høglund, "Autonomous Inspection of Wind Turbines and Buildings using an UAV," Master's thesis, Norwegian University of Science and Technology, 2014. [Online]. Available: http://hdl.handle.net/11250/261287

[104] D. Jones, "Power Line Inspection - An UAV Concept," in *The IEE Forum on Autonomous Systems*, 2005.

[105] P. Campoy, P. J. Garcia, A. Barrientos, J. del Cerro, I. Aguirre, A. Roa, R. Garcia, and J. M. Muñoz, "An Stereoscopic Vision System Guiding an Autonomous Helicopter for Overhead Power Cable Inspection," in *International Workshop RobVis*, 2001.

[106] K. Máthé and L. Buşoniu, "Vision and Control for UAVs: A Survey of General Methods and of Inexpensive Platforms for Infrastructure Inspection," *Sensors*, vol. 15, pp. 14 887–14 916, 2015.

[107] L. Marconi, F. Basile, G. Caprari, R. Carloni, P. Chiacchio, C. Hurzeler, V. Lippiello, R. Naldi, J. Nikolic, B. Siciliano, S. Stramigioli, and E. Zwicker, "Aerial Service Robotics: the AIRobots Perspective," in *International Conference on Applied Robotics for the Power Industry*, 2012, pp. 64–69.

[108] A. E. Jimenez-Cano, J. Braga, G. Heredia, and A. Ollero, "Aerial Manipulator for Structure Inspection by Contact from the Underside," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 1879–1884.

[109] K. Alexis, G. Darivianakis, M. Burri, and R. Siegwart, "Aerial Robotic Contact-based Inspection: Planning and Control," *Autonomous Robots*, vol. 40, no. 4, pp. 631–655, 2016.

[110] J. Cacace, A. Finzi, V. Lippiello, G. Loianno, and D. Sanzone, "Aerial Service Vehicles for Industrial Inspection: Task Decomposition and Plan Execution," *Applied Intelligence*, vol. 42, no. 1, pp. 49–62, 2015.

[111] H. Wu, M. Lv, C. A. Liu, and C. Y. Liu, "Planning Efficient and Robust Behaviors for Model-based Power Tower Inspection," in *International Conference on Applied Robotics for the Power Industry*, 2012, pp. 163–166.

[112] A. Santamaria-Navarro and J. Andrade-Cetto, "Hierarchical Task Control for Aerial Inspection," in *euRathlon-ARCAS Workshop and Summer School on Field Robotics*, 2014.

[113] D. Fox, W. Burgard, and S. Thrun, "The Dynamic Window Approach to Collision Avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[114] R. T. Chin and C. A. Harlow, "Automated Visual Inspection: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 4, no. 6, pp. 557–573, 1982.

[115] T. S. Newman, "A Survey of Automated Visual Inspection," *Computer Vision and Image Understanding*, vol. 61, no. 2, pp. 321–262, 1995.

[116] E. N. Malamas, E. G. M. Petrakis, M. Zervakis, L. Petit, and J.-D. Legat, "A Survey on Industrial Vision Systems, Applications and Tools," *Image and Vision Computing*, vol. 21, pp. 171–188, 2003.

[117] X. Xie, "A Review of Recent Advances in Surface Defect Detection using Texture Analysis Techniques," *Electronic Letters on Computer Vision and Image Analysis*, vol. 7, no. 3, pp. 1–22, 2008.

[118] T. Yamaguchi and S. Hashimoto, "Fast Crack Detection Method for Large-size Concrete Surface Images using Percolation-based Image Processing," *Machine Vision and Applications*, vol. 21, no. 5, pp. 797–809, 2010.

[119] M. R. Jahanshahi, J. S. Kelly, S. F. Masri, and G. S. Sukhatme, "A Survey and Evaluation of Promising Approaches for Automatic Image-based Defect Detection of Bridge Structures," *Structure and Infrastructure Engineering*, vol. 5, no. 6, pp. 455–486, 2009.

[120] M. Mumtaz, A. B. Masoor, and H. Masood, "A New Approach to Aircraft Surface Inspection based on Directional Energies of Texture," in *International Conference on Pattern Recognition*, 2010, pp. 4404–4407.

[121] T. Amano, "Correlation Based Image Defect Detection," in *International Conference on Pattern Recognition*, 2006, pp. 163–166.

[122] H. Peres Castilho, J. R. Caldas Pinto, and A. Limas Serafim, "NN Automated Defect Detection Based on Optimized Thresholding," in *International Conference on Image Analysis and Recognition*, 2006, pp. 790–801.

[123] H. Jia, Y. L. Murphey, J. Shi, and T.-S. Chang, "An Intelligent Real-time Vision System for Surface Defect Detection," in *International Conference on Pattern Recognition*, vol. III, 2004, pp. 239–242.

[124] J. Canny, "A Computational Approach To Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[125] I. Abdel-Qader, O. Abudayyeh, and M. E. Kelly, "Analysis of Edge-Detection Techniques for Crack Identification in Bridges," *Journal of Computing in Civil Engineering*, vol. 17, no. 4, pp. 255–263, 2003.

[126] L. Meng, Z. Wang, Y. Fujikawa, and S. Oyanagi, "Detecting Cracks on a Concrete Surface using Histogram of Oriented Gradients," in *International Conference on Advanced Mechatronic Systems*, 2015, pp. 103–107.

[127] Y. Fujita and Y. Hamamoto, "A Robust Automatic Crack Detection Method for Noisy Concrete Surfaces," *Machine Vision and Applications*, vol. 22, no. 2, pp. 245–254, 2011.

[128] Y. Fujita, Y. Mitani, and Y. Hamamoto, "A Method for Crack Detection on a Concrete Structure," in *International Conference on Pattern Recognition*, 2006.

[129] T. Fawcett, "An Introduction to ROC Analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[130] P. Subirats, J. Dumoulin, V. Legeay, and D. Barba, "Automation of Pavement Surface Crack Detection using the Continuous Wavelet Transform," in *IEEE International Conference on Image Processing*, 2006, pp. 3037–3040.

[131] S.-N. Yu, J.-H. Jang, and C.-S. Han, "Auto Inspection System using a Mobile Robot for Detecting Concrete Cracks in a Tunnel," *Automation in Construction*, vol. 16, no. 3, pp. 255–261, 2007.

[132] S. Chambon, P. Subirats, and J. Dumoulin, "Introduction of a Wavelet Transform based on 2D Matched Filter in a Markov Random Field for Fine Structure Extraction: Application on Road Crack Detection," in *IS&T/SPIE Electronic Imaging - Image Processing: Machine Vision Applications II*, 2009.

[133] M. Nieniewski, L. Chmielewski, A. Jóźwik, and M. Skłodowski, "Morphological Detection and Feature-based Classification of Cracked Regions in Ferrites," *Machine Graphics and Vision*, vol. 8, no. 4, pp. 699–712, 1999.

[134] W. Zhang, Z. Zhang, D. Qi, and Y. Liu, "Automatic Crack Detection and Classification Method for Subway Tunnel Safety Monitoring," *Sensors*, vol. 14, pp. 19 307–19 328, 2014.

[135] G. B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme Learning Machine for Regression and Multiclass Classification," *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.

[136] N. Tanaka and K. Uematsu, "A Crack Detection Method in Road Surface Images Using Morphology," in *IAPR Workshop on Machine Vision Applications*, 1998, pp. 154–157.

[137] H. Zheng, L. X. Kong, and S. Nahavandi, "Automatic Inspection of Metallic Surface Defects using Genetic Algorithms," *Journal of Materials Processing Technology*, vol. 125-126, pp. 427–433, 2002.

[138] M. Yoshioka and S. Omatu, "Defect Detection Method using Rotational Morphology," *Artificial Life and Robotics*, vol. 14, no. 1, pp. 20–23, 2009.

[139] S. H. Cho, K. Hisatomi, and S. Hashimoto, "Cracks and Displacement Feature Extraction of the Concrete Block Surface," in *IAPR Workshop on Machine Vision Applications*, 1998, pp. 246–249.

[140] T. Yamaguchi and S. Hashimoto, "Image Processing based on Percolation Model," *IEICE Transactions on Information and Systems*, vol. 89, no. 7, pp. 2044–2052, 2006.

[141] Z. Qu, L.-D. Lin, Y. Guo, and N. Wang, "An Improved Algorithm for Image Crack Detection based on Percolation Model," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 10, no. 2, pp. 214–221, 2015.

[142] S. Sorncharean and S. Phiphobmongkol, "Crack Detection on Asphalt Surface Image using Enhanced Grid Cell Analysis," in *IEEE International Symposium on Electronic Design, Test and Applications*, 2008.

[143] S. Avril, A. Vautrin, and Y. Surrel, "Grid Method: Application to the Characterization of Cracks," *Experimental Mechanics*, vol. 44, no. 1, pp. 37–43, 2004.

[144] R. Oullette, M. Browne, and K. Hirasawa, "Genetic Algorithm Optimization of a Convolutional Neural Network for Autonomous Crack Detection," in *Congress on Evolutionary Computation*, 2004, pp. 516–521.

[145] L. Zhang, F. Yang, Y. D. Zhang, and Y. J. Zhu, "Road Crack Detection using Deep Convolutional Neural Network," in *IEEE International Conference on Image Processing*, 2016, pp. 3708–3712.

[146] Y. Freund and R. E. Schapire, "A Short Introduction to Boosting," *Journal of Japanese Society for Artificial Intelligence*, vol. 14, no. 5, pp. 771–780, 1999.

[147] G. Zhao, T. Wang, and J. Ye, "Anisotropic Clustering on Surfaces for Crack Extraction," *Machine Vision and Applications*, vol. 26, no. 5, pp. 675–688, 2015.

[148] M. R. Jahanshahi, S. F. Masri, C. W. Padgett, and G. S. Sukhatme, "An Innovative Methodology for Detection and Quantification of Cracks Through Incorporation of Depth Perception," *Machine Vision and Applications*, vol. 24, no. 2, pp. 227–241, 2013.

[149] K. N. Snavely, "Scene reconstruction and visualization from internet photo collections," Ph.D. dissertation, University of Washington, 2008.

[150] S. Ghanta, T. Karp, and S. Lee, "Wavelet Domain Detection of Rust in Steel Bridge Images," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2011, pp. 1033–1036.

[151] I. Jolliffe, *Principal Component Analysis*. Springer, 2002.

[152] M. R. Jahanshahi and S. F. Masri, "Effect of Color Space, Color Channels, and Sub-Image Block Size on the Performance of Wavelet-based Texture Analysis Algorithms: An Application to Corrosion Detection on Steel Structures," in *Computing in Civil Engineering*, 2013, pp. 685–692.

[153] K. Y. Choi and S. S. Kim, "Morphological Analysis and Classification of Types of Surface Corrosion Damage by Digital Image Processing," *Corrosion Science*, vol. 47, no. 1, pp. 1–15, 2005.

[154] H. F. Kaiser, "The Varimax Criterion for Analytic Rotation in Factor Analysis," *Psychometrika*, vol. 23, no. 3, pp. 187–200, 1958.

[155] M. P. Bento, F. N. S. de Medeiros, I. C. de Paula Jr., and G. L. B. Ramalho, "Image Processing Techniques Applied for Corrosion Damage Analysis," in *Brazilian Symposium on Computer Graphics and Image Processing*, 2009.

[156] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural Features for Image Classification," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 3, no. 6, pp. 610–621, 1973.

[157] T. Kohonen, *Self-Organizing Maps*. Springer, 2001.

[158] F. N. S. Medeiros, G. L. B. Ramalho, M. P. Bento, and L. C. L. Medeiros, "On the Evaluation of Texture and Color Features for Nondestructive Corrosion Detection," *EURASIP Journal on Advances in Signal Processing*, 2010.

[159] A. R. Webb and K. D. Copsey, *Statistical Pattern Recognition*, 3rd ed. Wiley, 2011.

[160] M. Yamana, H. Murata, T. Onoda, T. Ohashi, and S. Kato, "Development of System for Crossarm Reuse Judgment on the Basis of Classification of Rust Images using Support Vector Machine," in *IEEE International Conference on Tools with Artificial Intelligence*, 2005.

[161] F. Tsutsumi, H. Murata, T. Onoda, O. Oguri, and H. Tanaka, "Automatic Corrosion Estimation using Galvanized Steel Images on Power Transmission Towers," in *Transmission and Distribution Conference and Exposition: Asia and Pacific*, 2009.

[162] G. Ji, Y. Zhu, and Y. Zhang, "The Corroded Defect Rating System of Coating Material Based on Computer Vision," in *Transactions on Edutainment VIII*, vol. LNCS 7220, 2012, pp. 210–220.

[163] L. Vincent and P. Soille, "Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 583–598, 1991.

[164] S. A. Idris, F. A. Jafar, and S. Saffar, "Improving Visual Corrosion Inspection Accuracy with Image Enhancement Filters," in *International Conference on Ubiquitous Robots and Ambient Intelligence*, 2015, pp. 129–132.

[165] L. Petricca, T. Moss, G. Figueroa, and S. Broen, "Corrosion Detection using A.I.: A Comparison of Standard Computer Vision Techniques and Deep Learning Model," in *International Conference on Computer Science, Engineering and Information Technology*, 2016, pp. 91–99.

[166] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Neural Information Processing Systems*, 2012.

[167] G. Cheng and A. Zelinsky, "Supervised Autonomy: A Framework for Human-Robot Systems Development," *Autonomous Robots*, vol. 10, pp. 251–266, 2001.

[168] S. Shen, N. Michael, and V. Kumar, "Autonomous Indoor 3D Exploration with a Micro-Aerial Vehicle," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 9–15.

[169] P.-J. Bristeau, F. Callou, D. Vissiere, and N. Petit, "The Navigation and Control Technology Inside the Ar.Drone Micro UAV," in *IFAC World Congress*, 2011, pp. 1477–1484.

[170] A. Briod, J.-C. Zufferey, and D. Floreano, "Optic-Flow based Control of a 46g Quadrotor," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[171] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer, "2D Simultaneous Localization And Mapping for Micro Air Vehicles," in *European Micro Aerial Vehicles*, 2006.

[172] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart, "Onboard IMU and Monocular Vision Based Control for MAVs in Unknown In- and Outdoor Environments," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 3056–3063.

[173] J. J. Engel, "Autonomous Camera-based Navigation of a Quadrocopter," Master's thesis, Technical University of Munich, 2011.

[174] D. Scaramuzza, M. C. Achtelik, L. Doitsidis, F. Friedrich, E. B. Kosmatopoulos, A. Martinelli, M. W. Achtelik, M. Chli, S. A. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G. H. Lee, S. Lynen, L. Meier, M. Pollefeys, A. Renzaglia, R. Siegwart, J. C. Stumpf, P. Tanskanen, C. Troiani, and S. Weiss, "Vision-Controlled Micro Flying Robots: from System Design to Autonomous Navigation and Mapping in GPS-denied Environments," *IEEE Robotics and Automation Magazine*, vol. 21, no. 3, pp. 26–40, 2014.

[175] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, "Stereo Vision and Laser Odometry for Autonomous Helicopters in GPS-denied Indoor Environments," in *SPIE Unmanned Systems Technology XI*, 2009.

[176] K. Schauwecker, "Stereo Vision for Autonomous Micro Aerial Vehicles," Master's thesis, University of Tübingen, 2014.

[177] F. Golnaraghi and B. C. Kuo, *Automatic Control Systems*, Ninth ed. Wiley, 2010.

[178] R. C. Arkin, *Behavior-based Robotics*. MIT Press, 1998.

[179] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[180] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys, "An Open Source and Open Hardware Embedded Metric Optical Flow CMOS Camera for Indoor and Outdoor Applications," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 1736–1741.

[181] M. Ruffo, M. D. Castro, L. Molinari, R. Losito, A. Masi, J. Kovermann, and L. Rodrigues, "New Infrared Time-of-flight Measurement Sensor for Robotic Platforms," in *IMEKO TC4 Int. Symposium and Int. Workshop on ADC Modelling and Testing*, 2014, pp. 13–18.

[182] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an Open-Source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[183] A. Censi, "An ICP Variant using a Point-to-Line Metric," in *IEEE International Conference on Robotics and Automation*, 2008.

[184] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[185] G. Grisetti, C. Stachniss, and W. Burgard, "Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters," *IEEE Transactions on Robotics*, vol. 23, pp. 34–46, 2007.

[186] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Wiley, 2000.

[187] A. Ortiz and G. Oliver, "On the Use of the Overlapping Area Matrix for Image Segmentation Evaluation: A Survey and New Performance Measures," *Pattern Recognition Letters*, vol. 27, no. 16, pp. 1916–1926, 2006.

[188] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 3rd ed. Academic Press, 2006.

[189] K. I. Law, "Texture Energy Measures," in *Image Understanding Workshop*, 1979, pp. 47–51.

[190] K. I. Law, "Textured Image Segmentation," Image Processing Institute, University of Southern California, Tech. Rep. 940, 1980.

[191] J. Friedman, T. Hastie, and R. Tibshirani, "Additive Logistic Regression: A Statistical View of Boosting," *The Annals of Statistics*, vol. 38, no. 2, pp. 337–374, 2000.

[192] P. Viola and M. Jones, "Robust Real-Time Face Detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[193] A. Vezhnevets and V. Vezhnevets, "Modest AdaBoost - Teaching AdaBoost to Generalize Better," Graphicon, Tech. Rep., 2005.

[194] B. K. P. Horn, *Robot Vision*. MIT Press, 1986.

[195] C. Tomasi and R. Manduchi, "Bilateral Filtering for Gray and Color Images," in *International Conference on Computer Vision*, 1998, pp. 839–846.

[196] A. Borji and L. Itti, "State-of-the-Art in Visual Attention Modeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 185–207, 2013.

[197] C. Koch and S. Ullman, "Shifts in Selective Visual Attention: Towards the Underlying Neural Circuitry," *Human Neurobiology*, vol. 4, no. 4, pp. 219–227, 1985.

[198] J. M. Wolfe, "Integrated Models of Cognitive Systems," in *Guided Search 4.0*, W. D. Gray, Ed. Oxford University Press New York, NY, 2007, ch. 8, pp. 99–119.

[199] T. Avraham and M. Lindenbaum, "Esaliency (Extended Saliency): Meaningful Attention using Stochastic Image Modeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 693–708, 2010.

[200] A. Borji, M. N. Ahmadabadi, and B. N. Araabi, "Cost-sensitive Learning of Top-Down Modulation for Attentional Control," *Machine Vision and Applications*, vol. 22, no. 1, pp. 61–76, 2011.

[201] J. Li, Y. Tian, T. Huang, and W. Gao, "Probabilistic Multi-task Learning for Visual Saliency Estimation in Video," *International Journal of Computer Vision*, vol. 90, no. 2, pp. 150–165, 2010.

[202] L. Zhang, B. Qiu, X. Yu, and J. Xu, "Multi-scale Hybrid Saliency Analysis for Region of Interest Detection in Very High Resolution Remote Sensing Images," *Image and Vision Computing*, vol. 35, pp. 1–13, 2015.

[203] L. Itti, C. Koch, and E. Niebur, "A Model of Saliency-based Visual Attention for Rapid Scene Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, nov 1998.

[204] G. Kootstra, A. Nederveen, and B. D. Boer, "Paying Attention to Symmetry," in *British Machine Vision Conference*. BMVA Press, 2008, pp. 111.1–111.10.

[205] D. Reisfeld, H. Wolfson, and Y. Yeshurun, "Context-Free Attentional Operators: The Generalized Symmetry Transform," *International Journal of Computer Vision*, vol. 14, pp. 119–130, 1995.

[206] G. Kootstra and L. R. B. Schomaker, "Prediction of Human Eye Fixations using Symmetry," in *Annual Conference of the Cognitive Science Society*, 2009, pp. 56–61.

[207] L. Zhang, M. H. Tong, T. K. Marks, H. Shan, and G. W. Cottrell, "SUN: A Bayesian Framework for Saliency using Natural Statistics," *Journal of Vision*, vol. 8, no. 7, pp. 1–20, 2008.