A PROPOSAL TO ESTIMATE THE MOTION OF AN UNDERWATER VEHICLE THROUGH VISUAL MOSAICKING

Rafael GARCÍA CAMPOS

ISBN: 84-8458-124-1
Dipòsit legal: GI-70-2002
http://hdl.handle.net/10803/7716

A Proposal to Estimate the Motion of an Underwater Vehicle through Visual Mosaicking

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRONICS, INFORMATICS AND AUTOMATION AND THE COMMITTEE ON GRADUATE STUDIES
OF THE UNIVERSITY OF GIRONA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY (ENGINEERING)

by

Rafael Garcia

Director: Xavier Cufí

October 2001

A la Yolanda, qui m'ha ensenyat a estimar.

Agraïments

D’alguna manera, mentre escric aquestes línies, m’envaeix la satisfacció de tenir la feina feta. Una feina que no hauria estat possible sense l’ajuda de molta gent. Tinc la sensació que estem a punt de tancar una porta que ens fa deixar enrera moltes vivències... i que ens obliga a seguir el camí cap endavant. I dic “estem”, perquè tinc la certesa de que aquest camí no l’he recorregut tot sol. És per això que vull agrair al *Gran Mestre* Xevi no només haver marcat el camí que havíem de seguir, sinó també haver-me agafat de la mà en els moments difícils, ensenyant-me a afrontar els problemes amb seriositat, rigor i sense perdre el bon humor. També vull agrair al nostre *Cap de Colla*, en JoanB, per haver-me brindat l’oportunitat de treballar en el mon submarí, ple d’encants i de misteris; i el fet d’haver-nos aconsellat en tot moment per fer-nos veure més enllà. Sense l’aportació dels seus coneixements i ànims constants aquest treball no seria el que és.

I què dir del “quinteto maravillas”, Perico-Quimet-PepF-JordiF-JoanM, que ha hagut d’actuar darrerament fins a les tantes de la matinada perquè aquest treball veiés la llum, davant un públic ensopit format per pizzes i cafès. A tots ells els dec molt. El *cap d’expedició* Pere per les anades i vingudes cap a Ca la Margarida per anar a banyar l’URIS, el *domador d’equacions* Quim per haver patit en les seves carns les “quatre equacions” que surten en aquesta tesi, en *NuncaTercerosChaptersFueron-Buenos* PepF per haver lluitat airosament contra el cardenal Richard-lieu, fins fer-lo exiliar a Anglaterra, i els *TextureMen* Frexi i JoanM per haver comès l’error de treballar amb textures abans que jo: equivocació que no es paga amb un parell d’hores (com ara ja sabeu). Moltes gràcies a tots.

També vull agrair a la gent de la UPC, especialment en Josep Amat i l'Alícia pels ànims que m'heu donat "*amb això tens una bona peça al teler*", i la vostra accessibilitat amb tot allò referent a la tesi. Sempre és reconfortant saber que veieu amb bons ulls la feina que fem.

Agraeixo també l'esforç i el suport que he rebut de la resta de membres del Grup de Visió per Computador i Robòtica, començant per en Mangui, que ha estat vital en el seu servei d'atenció a l'usuari de Matlab les 24 hores, en Jordi Pagès pel suport logístic "*no deixis que s'acabi el paper*", l'Ela, la Jessi, en XMuno i en Lladow per haver-vos fet anar de bòlid amb "*necessito una Meteor funcionant*", en Pacheco, en Magí, en Tomàs i en Toni pel seu suport i disponibilitat constant, i el "hawaiian boy" d'en MarcC a qui hem passat més d'una patata calenta. I vull donar les gràcies a les secres per haver-me fet riure quan més ho necessitava i haver-se preocupat per la meva esquena com si fos seva. I en Pop pels seus ànims constants, i per aconseguir que els altres tinguéssim la gratificant sensació de tenir poca feina, veient el seu desbordament descomunal.

També vull agrair d'una manera molt especial l'ajuda dels meus "nens" més recents. En Ritu "*a veure torna-m'ho a explicar, que aquest cap de setmana se m'ha complicat molt*", en JordiN "*si, això també ho he provat*", en JordiP "*he trobat un llibre del 50 on això està explicat*", en Pio "*tinc una llibreria que et deixarà parat*". Durant els darrers temps heu aconseguit imprimir-me la il·lusió i la joventut per tirar endavant amb ganes de riure.

I no voldria oblidar els "nens" que ja s'han independitzat i han marxat de casa. Miki, Xavi Pinsach, Carles Bosch, Tomás Medel. He après moltes coses amb vosaltres, i espero continuar sabent de vosaltres en el futur...

També gràcies a tots els companys de "la" DEIA (em pregunto quan es farà un home?). Han sigut moltes les mostres de suport del Departament en aquests darrers mesos. I a la colla de sempre: en Gassi, en Xevi, en Fidi, en Tito, en Ferran, en Santi i l'Albert. La veritat és que heu patit aquesta tesi tant com jo.

Vull agrair als meus pares per haver sabut portar-me fins aquí com la cosa més normal del mon, i per haver-me donat suport en cada passa que he fet no només en els estudis, sinó també en la vida. I gràcies a les meves germanes per haver-vos preocupat de mi com ho heu fet, i per actuar com a estimulant per fer-me arribar fins al final.

I “last but not least” dubto que aquesta tesi hagués vist la llum sense el suport incondicional de la meva “doctora” particular. Sense cap mena de dubte, ella ha sigut qui millor m’ha entès, qui m’ha mimat i qui m’ha estimat en aquest llarg viatge. Seria injust no reconèixer que tenir-la al costat fa que els moments més difícils es converteixin en un passeig de diumenge. És per això que ella ha de rebre el més profund dels meus agraïments.

P.D.- Ah! Per últim, vull tenir un reconeixement per l’anònim Telepizzero que tantes vegades ha tingut la paciència de picar el timbre de l’entrada i esperar que vingués el segurata per obrir la porta del PII. I total per vint duros de propina (que vindrien a ser... uhmmm... uns euros, a quan diuen que està?)

Contents

List of Tables

List of Figures

Glossary

This list summarizes the acronyms used in this thesis

ASKF	Augmented State Kalman Filter
AUV	Autonomous Underwater Vehicle
AUVS	Autonomous Underwater Vehicle Simulator
CML	Concurrent Mapping and Localization
COM	Co-Occurrence Matrix
CS	Correlation Score
CV	Characterization Vector
DGPS	Differential Global Positioning System
DOF	Degrees of Freedom
DVL	Doppler Velocity Log
EF	Energy Filters
EKF	Extended Kalman Filter
FOG	Fiber Optics Gyro
GPS	Global Positioning System
INS	Inertial Navigation System
KF	Kalman Filter

LBL	Long Baseline
LBP	Local Binary Patterns
MMSE	Minimum Mean Square Error
MRF	Markov Random Field
MSE	Mean Square Error
RLG	Ring Laser Gyro
RLS	Recursive Least Squares
ROV	Remotely Operated Vehicle
SBL	Short Baseline
SM	Similarity Measure
SVD	Singular Value Decomposition
USBL	Ultra-Short Baseline
UUV	Unmanned Underwater Vehicle

Nomenclature

The conventions which have been adopted throughout this thesis are summarized here.

General typographical conventions that apply to all mathematical objects:

Typographical Conventions		
f	Lowercase italic	Scalar
$\mathbf{m}$	Lowercase boldface	Vector
$\tilde{\mathbf{m}}$	Lowercase boldface with a tilde	Vector expressed in homogeneous coordinates
$\mathbf{H}$	Uppercase boldface	Stacked vector or matrix

Conventions used to describe the Kalman Filter equations:

Kalman Filtering Conventions	
$\mathbf{x}$	System State vector
$\mathbf{u}$	Input to the system (without noise)
$\mathbf{z}$	Measure vector
$\mathbf{w}$	System white Gaussian noise vector (~N(0,Q))
$\mathbf{v}$	Measure white Gaussian noise vector (~N(0,R))
$\mathbf{\Phi}$	Propagation/state evolution matrix
$\mathbf{G}$	Input matrix
$\mathbf{H}$	Measurement Matrix
$\mathbf{B}$	System Noise Matrix
$\mathbf{Q}$	System Noise Covariance Matrix
$\mathbf{R}$	Measurement Noise Covariance Matrix

Chapter 1

Introduction

Bio-inspired systems try to emulate the behavior of those systems that are found in the nature. From this statement we could wonder why fishes have eyes. What do they do with them? Which behaviors are being accomplished by means of the sense of “looking at things”? It is obvious that image sensing helps living systems to interact with their environment. Images are a rich source of information that can also be exploited in non-favorable environments such as the underwater scenario.

When we try to emulate the sense of vision by means of a computer system, we are entering the terrain of the *Computer Vision* field. Computer vision can be defined as the process of extracting relevant information of the physical world from images, using a computer to obtain this information [Mar82]. Indeed, this information can be of different nature, going from measuring some statistical properties of an industrial part to detecting possible difficulties in the surrounding area to allow autonomous vehicle navigation.

1.1 Underwater robotics

In the past years, the Unmanned Underwater Vehicles (UUVs) that have almost always been used are the so-called Remotely Operated Vehicles (ROVs). That is, a pilot/operator remotely controls the vehicle from a mother vessel. The connection between the vessel and the robot could be an umbilical tether or an ultrasound link. Both approaches have their advantages and drawbacks. While the umbilical provides both control signals and power, it is affected by currents and difficults the vehicle motion. On the other hand, ultrasound links can send the pertinent control signals and are not affected by currents, but leave the robot on its own concerning the power supply. This fact implies a reduction in the duration of the mission, limited by the robot autonomy.

The second type of UUVs that are starting to appear in real missions are the Autonomous Underwater Vehicles (AUVs). These submersibles take their own decisions within the mission based on the readings of the on-board sensors and their control architecture, without the need of a human operator. Currently, most AUVs missions are limited to engineering demonstrations to prove new technologies and perform simpler tasks than those performed by ROVs.

Typical missions for ROVs and AUVs include inspection of submersed structures, sea-floor exploration, measurements within the water column, wreck discovery and localization, pipe and cable inspection, etc. Independently of the mission, it is highly important for all UUVs to incorporate the capability of positioning and localizing themselves in the underwater environment. Once the positioning problem is solved, the vehicle can perform navigation by means of achieving given positions through the adequate path planning algorithms. These capabilities are not only necessary for AUVs, but also for ROVs. Consider for example a typical survey mission, where the vehicle has to follow a straight trajectory, or to keep station. The ROV operator should constantly maneuver the vehicle with joysticks to perform these low-level tasks. This is a very tedious operation, especially in presence of undersea currents, and the success of the mission depends on the skills and experience of the operator to control the vehicle. while the scientist provides mission-level directives to the pilot. For this reason it is desirable that these low-level operations be performed directly by the robot, allowing the operator to concentrate on the high-level tasks.

1.2 Computer vision as a positioning tool

As already stated, all UUV missions require some method to sense vehicle position and localization. Normally, most teleoperated vehicles incorporate a video camera to provide a visual feedback to the robot operator. From this viewpoint, the idea of using this sensor for positioning the vehicle is very attractive and cost-effective. The following sections compare the different positioning sensors with computer vision.

1.2.1 Definition of the Positioning Problem

Let $\{B\}$ be an earth fixed frame and $\{G\}$ a robot-fixed frame. Solving the positioning problem for an underwater vehicle in 6 degrees of freedom (DOF) means estimating the position vector $\eta=(x,y,z,\phi,\theta,\psi)^T$ where x, y, z define the robot position with respect to the earth fixed frame and ϕ, θ, ψ, are the rotation angles with respect to x, y and z robot axis.

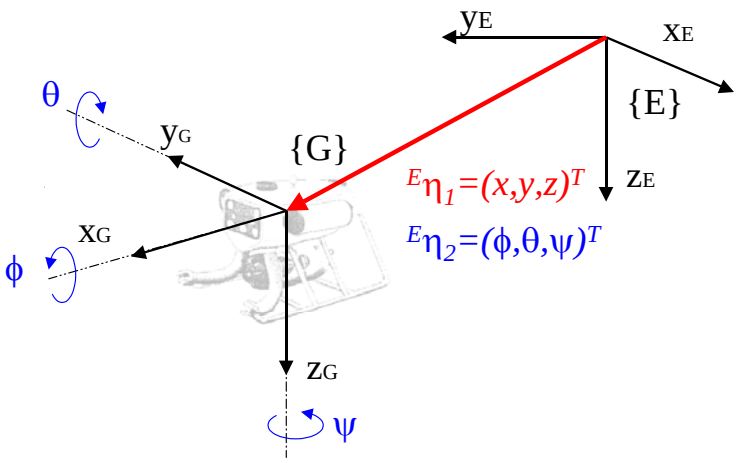


Figure 1.1. Robot position and attitude.

1.2.2 Positioning sensors

It is possible to find a range of sensing technologies to provide the vehicle with knowledge about its position: Doppler Velocity Log (DVL), Acoustic Transponder Networks, Inertial Navigation Systems (INS) and Global Positioning System (GPS). Those sensors are briefly analyzed below, paying special attention to the aspects of accuracy, easy of use, limitations and cost.

Compasses and Inclinometers: *Compasses* measure orientation with respect to the Earth's magnetic field. These sensors are inexpensive and drift-free, although susceptible to local variations in the ambient magnetic field. When mounted onboard, their response uses to be non-linear. Nevertheless it can be easily linearized by means of deviation tables which can be obtained through a calibration method. An *inclinometer* is a device that measures the orientation of the gravity vector, thus it measures deviations from the gravity frame. The underwater vehicle can take profit of inclinometers to sense variations in pitch and roll due to unexpected depressions in its environment.

(a) (b)

Figure 1.2. Photograph of two commercial INS: (a) ISIS; (b) TriPos.

INS: Inertial Navigation Systems are an integrated system (see Figure 1.2) composed by *gyros* and *accelerometers*. Gyros measure angular velocity by exploiting basic Newtonian mechanics. The first form of this device was a rapidly spinning mass, having a strong kinetic momentum and being suspended in a gimbal. Because angular momentum is conserved, any attempt to change the orientation of the gyroscope results in an effective force that would lead to precession. Measuring this force determines the angular velocity of the gyro. Although mechanical gyros based on this principle can achieve a very good accuracy (bias error $\cong 0.015^{o}$/h), they are big, heavy and expensive (above 10.000$), so they are not suitable for small low cost underwater robots. Nowadays gyroscopic sensors are based on Ring Laser Gyro (RLG) or Fiber Optics Gyro (FOG) technology avoiding moving parts [Law98]. Both technologies are based on the "Sagnac effect". The principle of function involves the determination of the phase shift between two counter-propagating light beams. For a RLG this occurs in an evacuated mirrored cavity, but in the FOG the same effect can be obtained in a fiber coil, obtaining a lower-cost solution. While RLG gyros are more accurate

(bias error ≅ 0.1-0.001º/h), FOG sensors are cheaper at cost of lower accuracy (bias error ≅ 10-0.01 º/h).

The principle of operation of *accelerometers* is based on the sensing of small accelerations in each of the three directional axis. Physically, accelerometers are spring-mounted masses whose displacement under acceleration can be measured from the spring-mass relation and the second Newton's Law. For low-speed underwater robots, the experimented acceleration is normally very small (≅ 0.1 m/s^2), hence only high accuracy accelerometers can be used.

An INS estimates the position by dead reckoning, as illustrated in Figure 1.3. The sensed acceleration ($^G\mathbf{a}$) is first integrated to get the lineal velocity ($^G\nu_1$) and then composed with the angular rated ($^G\nu_2$) sensed by the gyros to compose a 6 DOF velocity vector. Using a simple cinematic transformation $J(\eta)$ [Fos95], the rate of change of position and orientation with respect to frame *E* is computed and then integrated over time to get the position and orientation vector. This double integration quickly generates uncertain estimates. For a reasonable accuracy, both inertial systems are much too big and expensive to be used in a small UUV, while smaller and cheaper ones suffer from excessive drift to be used in real missions.

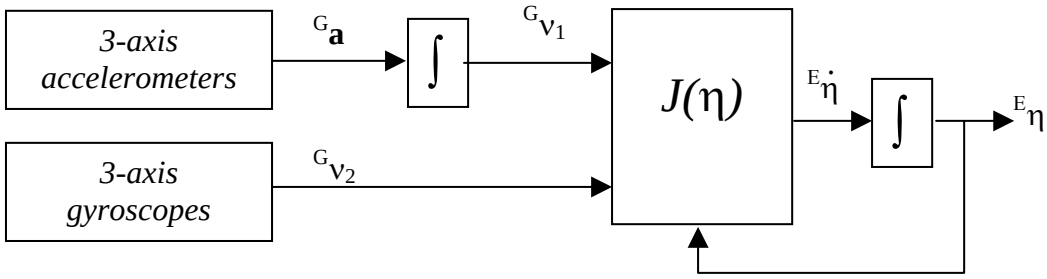


Figure 1.3. Position estimation by dead reckoning.

Sonar: It is the acronym for *SOund Navigation And Ranging*. The good propagation properties of sound waves through water has been exploited by a wide variety of devices to construct object detection systems. These devices are collectively known as sonar. Typically, a sonar system emits ultrasonic pulses and listens for reflected pulses from potential obstacles. Sonar equipment has been used to perform several tasks in the underwater environment, such as determination of

altitude above the sea-floor and multi-image surveys of the marine terrain. A serious limitation of the sonar systems is the multi-path problem, derived from the reflection of the acoustic signals from different objects. The speed of sound is dependent upon properties of the material through which it travels (see Table 1.1). Since it is an active sensor –the sensed energy is originally emitted by the device itself– it might not be adequate for some applications, *i.e.* tracking ultrasound-sensitive animals or military surveillance. Other problems of sonar systems include signal refraction, 3D spatial variation of the sound speed and different sources of noise (including the biological sources, the mother vessel or the submersible itself).

Table 1.1. Speed of sound in the air and water media

Medium	**Speed (meters/second)**
Air at 0°C	331 (increases with temperature and pressure)
Water	1400-1600 (increases with temperature, salinity, and depth)

Doppler Sonar: The operation of this device is based on the Doppler principle: the frequency of the received signal differs from the frequency of the emitted one when the source and the reference point are in motion relative to each other [Jor93]. This frequency is proportional to the relative velocity of motion. Many systems use 3 or 4 sonar beams. In the 4-beams configuration (*Janus configuration*) the transducers are located forward, backward, port and starboard, at right angles to each other. In the 3-beams configuration (Figure 1.4) they are located every 120º. In this way, three or four velocity vectors corresponding to the speed of the vehicle in every direction are obtained. The precision and accuracy of the Doppler sensor is a complex function which depends on various factors: altitude from the seabed, signal power, frequency, pulse length, velocity of sound accuracy and transducer alignment [Lar00a]. When the Doppler sonar is used far from the seabed, its measurements are highly inaccurate. There exists, hence, a range of altitudes where the sensor works efficiently. Low frequency Dopplers (300 Khz) can operate at altitudes up to 200 meters, while high frequency Dopplers can be operated at altitudes up to 30 meters. Errors in the assumed velocity of the sound are critical in the Doppler

equation. Unfortunately, the speed of sound is subject to considerable variations depending on the temperature, salinity and depth of the water medium [Wil60] (see Table 1.1). Therefore, the main drawbacks of this system are the variations of the speed of sound depending on the water conditions and the stationary drift, affecting more seriously to slow moving vehicles.

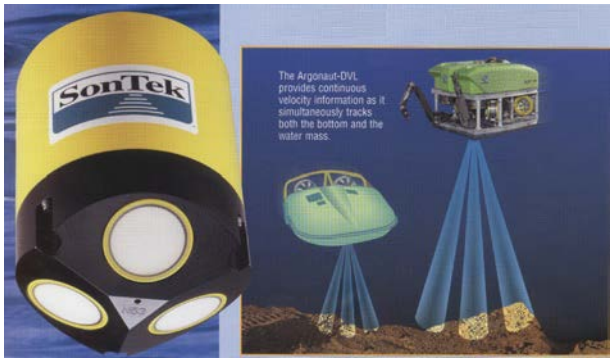
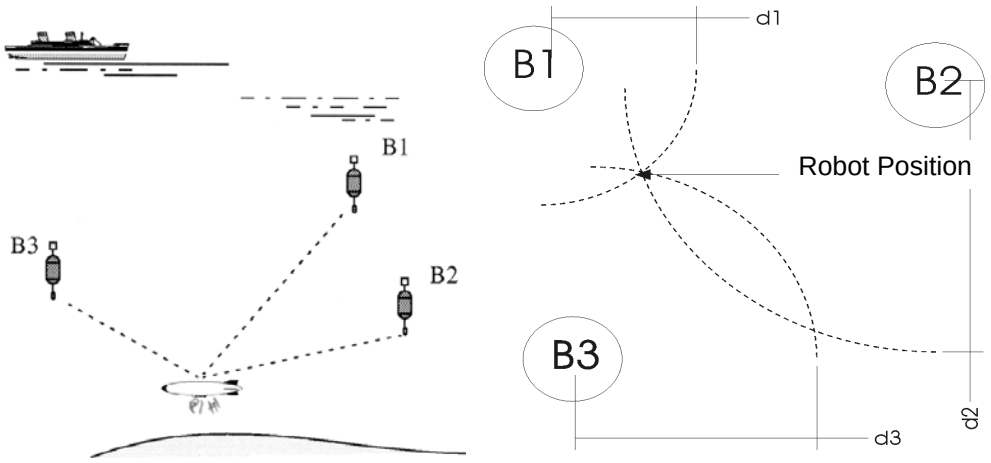


Figure 1.4. External aspect of the *SonTek* Doppler Velocity Log sensor for UUV applications.

Acoustic Transponder Networks: Various acoustic emitter/receiver devices can be placed covering the site of interest. Thus, accurate 3D positioning is achieved by navigating within the volume covered by this network. This positioning technology usually includes *long baseline* (LBL), *short baseline* (SBL) and *ultra-short baseline* (USBL) systems. Long Baseline consists of at least three transponders anchored in the water column, as shown in Figure 1.5. The underwater robot interrogates all transponders simultaneously using a given frequency. Each transponder replies using its own frequency. Once the underwater robot has received the responses, it computes their time of flight and estimates the *XY* position by triangulation. The update rates in position are typically from 1 to *n* seconds, depending on the area of influence of the LBL. For a distance of 750 meters, updates of 1 second are possible. This setup is a good solution for performing several missions in the same area [Lar00b]. The main drawback relies in the need of a careful calibration of the transponder position before the robot can operate. Short and ultra-short baselines systems do not require the need to deploy additional equipment in the water. For short baseline, two transponders are located in the mother-ship hull as far away as possible. The range measurements obtained from these transponders are

combined with the ship position estimated by the ship instrumentation, obtaining the robot position. On the other hand, ultra-short baseline requires only one transducer installed in the ship hull. This transducer provides three measurements: the slant distance and two bearing angles (horizontal and vertical). As in the case of the SBL, combining an USBL with the mother-ship positioning system, absolute position estimates are possible. System reliability is quite high and positioning area is also large. However, it has the disadvantage of the high operating cost.



(a) (b)

Figure 1.5. (a) Long Baseline Acoustic Transponder Networks (b) Detail of how position is computed by triangulation.

GPS: It is the acronym for *Global Positioning System*. It is a radio navigation system based on an array of 24 space-satellites that orbit the earth and some antennas for ground support. It provides three-dimensional position, velocity, and time, 24 hours a day, everywhere in the world and in all weather conditions. Unfortunately, GPS signals do not penetrate the water, so its use in underwater robotics is seriously limited. An alternative consists on equipping an UUV with a GPS system, and bring it to the surface at periodic intervals. In this way, the GPS signal can be used to reset accumulated drift errors in position that arose from the integration of inertial sensor measurements. In the case of Differential GPS (DGPS), the GPS signal is combined with a differential signal emitted by a

ground station located at a fixed position, thus improving the accuracy of the system. Position accuracies in the order of 1 to 2 meters can be obtained on the ocean surface with DGPS.

GPS-equipped sonar buoys: A good solution taking the best of both worlds is obtained integrating GPS and a network of sonar buoys [You91]. In this way, the UUV can estimate its position relative to the network of GPS receiver-equipped buoys. These buoys should also carry acoustic transponders to send sonar signals to the vehicle. Then, the UUV can obtain absolute 3D positioning from decoding the acoustic signals sent by the buoys and measuring the phase shifts. The accuracy does not match LBL, but the system is much easier to deploy. Some authors have referred to this system setup as "Underwater GPS" [An96,Tho97].

1.2.3 Computer Vision

Most of the UUVs which are being used nowadays in real sea missions are equipped with cameras to provide a comprehensive feedback to the robot operator and/or oceanic researchers. Vision is a natural choice in science-oriented missions that provides low-cost, high-rate and high-resolution information. Images are inherently intuitive to provide humans with a view of the underwater environment. Moreover, when this information is treated and processed adequately, it can be converted into position measurements.

A vision system for aiding UUV's navigation is composed of an on-board camera, a video digitizer, a host computer and, depending on the working depth, a source of light. Comparing these requirements with the sensors described above, it is obvious that Computer Vision emerges as a very low-cost sensing technology. Moreover, given the high resolution of digital imaging, measurement accuracy can be on the order of millimeters, while other sensing technologies obtain much less accurate resolutions. However, these measurements can only be used when the vehicle is close to the underwater terrain, and is limited by the visibility of the working area. Light attenuation in the water medium is one of the main problems the vision system has to face. Moreover, as the submersible increases the working depth, it has to carry artificial lights, increasing the power consumption of the vehicle.

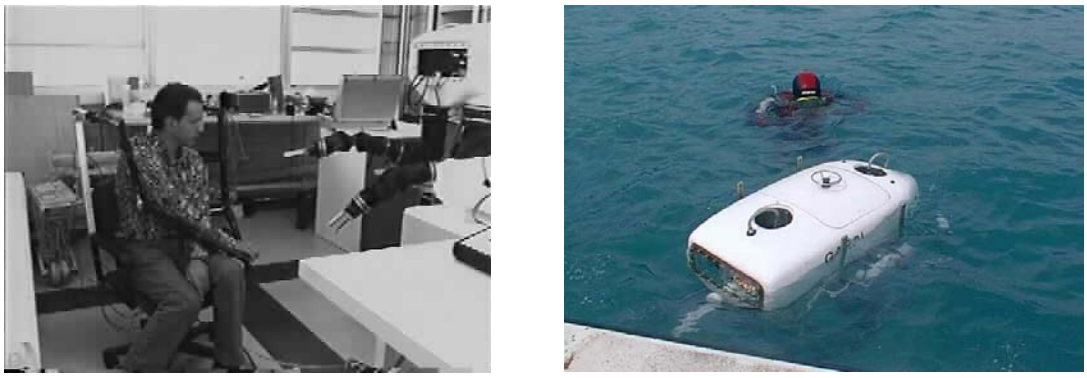
There exist a set of situations within a mission in which the UUV should remain at the same position for a certain period of time, such as inspection or welding

operations. Underwater currents or other disturbances obligate the vehicle to continuously correcting its position to keep station. In these situations, computer vision appears as a very good sensing technique, since its accuracy is not affected by any sort of drift as the mission time goes by. Although vision is an extremely accurate sensor for relative measurements, integration of these measurements to estimate absolute positions leads to error propagation problems as the vehicle moves.

In conclusion, cameras can provide good information in terms of resolution and actualization rate, but are limited to a very short range (much shorter than sonar) and in some cases can became “blind”. In ROV systems they provide a good interface to the user to teleoperate the robot. However, we should bear in mind that the different sensing systems that are equipping an UUV could be used together to map the ocean floor and, therefore, obtain position estimates. This **sensor fusion** means that the sensors can complement each other and give rise to a much more stable, robust and reliable system.

1.3 Experimental platforms for underwater exploration

The *Computer Vision and Robotics Group* of the University of Girona (UdG) has developed two underwater platforms to test different control and sensing strategies. The first underwater prototype was constructed in cooperation with the Polytechnical University of Catalonia (UPC) in the frame of project TAP 92-0792, funded by the Spanish CICYT organism. The aim of this project was to develop a remotely-operated underwater robot (GARBI), equipped with to arms to carry out simple manipulation tasks. From the results of this work, a second project, TAP95-0425-C02-02, consisted in introducing some improvements in the prototype and the tele-operational system. This project included the construction of an exoesqueleton, shown in Figure 1.6(a), to carry out the tele-operation of the arms. The system reproduces the movement of the operator’s arms making easier the manipulation of objects in the underwater environment. A camera located inside the robot provided the operator with a view of the robot’s environment. Figure 1.6(b) shows GARBI during a test mission.

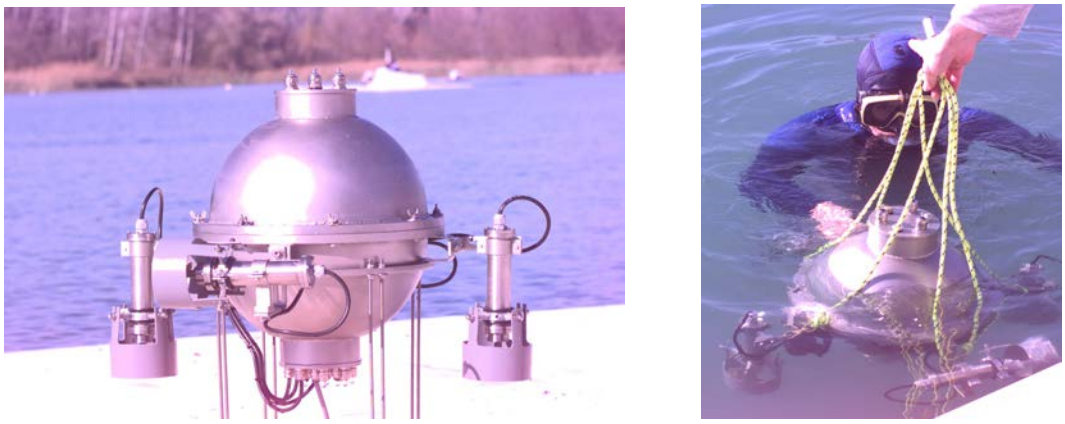


(a) (b)

Figure 1.6. (a) Exosqueleton used for the tele-operation of the arms;
(b) GARBI UUV in a sea mission.

A second prototype of GARBI was constructed within project MAR 97-0925-C02-02. This project consisted in studying the coordinated remote operation of both vehicles. Finally, the last CICYT project (MAR99-1062-C03-02), represents our first attempt to build a fully autonomous underwater vehicle. This is a joint project involving three universities: UdG, UPC and UIB (University of the Balear Islands). The main goal consists of building an AUV prototype called URIS[1] (see Figure 1.7), which will be able to execute simple missions like exploring an area while gathering information. Currently, the Computer Vision and Robotics Group plans to hand over to industry a simplified version of this vehicle which will be restricted to tele-operational tasks. The expected applications for the commercial version are observation of dams and collector walls, helping in rescue tasks, observation of the bottom of commercial and recreational ports, inspection of ship bottoms, and so on.

[1] URIS AUV is inspired on ODIN AUV developed in the Autonomous System Laboratory of the University of Hawaii at Manoa.



(a) (b)

Figure 1.7. (a) URIS on the dock at the Banyoles lake; (b) URIS being recovered.

Next, a brief introduction to the underwater vehicles designed and built by the Computer Vision and Robotics Group is given below. A further detailed description of these vehicles can be found in [Rid01].

The GARBI Underwater Vehicle

GARBI was conceived as a Remotely Operated Vehicle (ROV) for exploration in waters up to a depth of 200 meters. It was designed with the aim of building an underwater vehicle using low cost materials such as fibber-glass and epoxy resins. To solve the problem of resistance to underwater pressure, the vehicle is servo-pressurized to the external pressure by using a compressed air bottle, like those used in scuba diving. Air consumption is required only in the vertical displacements. When the robot dives in the heave direction, a servo system introduces air into the hull in order to increase the internal pressure until it reaches the external pressure. In addition, when the robot goes to the surface, decompression valves release the required amount of air to maintain the internal pressure at the same level as the external one (see Figure 1.8(b)). The vehicle can be equipped with two mechanical arms which would perform some simple tasks of object manipulation through remote-operation.

(a) (b)

Figure 1.8. (a) Structure of GARBI UUV. (b) GARBI UUV while surfacing.

The vehicle has 4 thrusters: two for horizontal movements (*x* axis) and two for vertical movements (*z* axis), as illustrated in Figure 1.8(a). It is possible to add a fifth thruster in the transverse direction (*y* axis). Due to the distribution of weight, the vehicle is passively stable in *roll* and *pitch*. The vehicle also has several sensors: 2 compasses, 2 pressure sensors and water speed and water leakage sensors. Dimensions are: 1.3 m length, 0.9 m height and 0.7 m width. Maximum speed is 1 knot and the weight is approximately 150 kg. Table 1.2 shows the principal characteristics of the GARBI UUV.

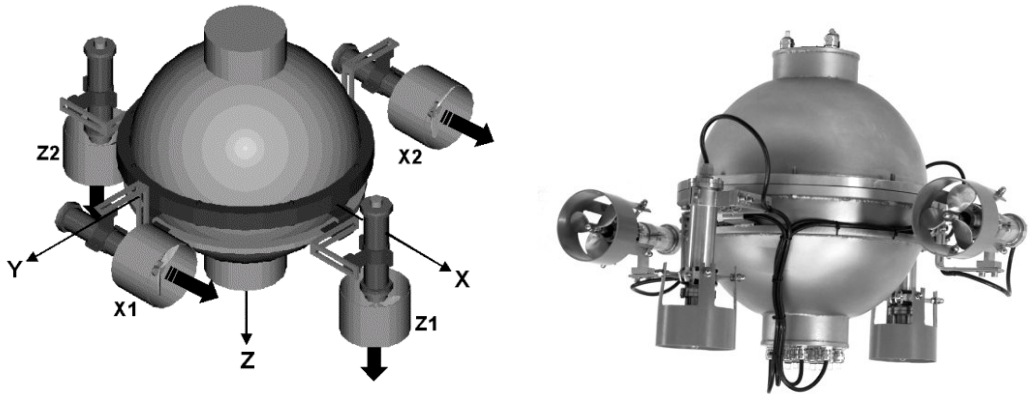
Table 1.2. Characteristics of GARBI.

Type	ROV
D.O.F.	4 (*x*,*y*,*z*,*Yaw*)
Stability	Passively stable in Roll and Pitch
Propulsion	4 thrusters (120 W approx.) 1 lateral thruster (optional)
Energy	DC power source Umbilical cable (video, power, ethernet)
Max. depth	200 meters
Sensors	2 Magnetic compasses (Yaw) 2 Pressure sensors (*z* and air bottle) Color Camera Water speed sensor Water leakage sensor
Other	2 arms with 3 DOF. Servo-pressurized.

An on-board i486 embedded computer is in charge of controlling the vehicle's sub-systems. The robot is connected to the surface station by means of an umbilical cable for power, video and Ethernet communication.

The URIS Underwater Vehicle

The URIS (*Underwater Robotic Intelligent System*) vehicle was designed with the aim of developing a small, light-weight, low-cost AUV to be used as a research testbed in a water tank testing facility. URIS minimizes the amount of operators which are necessary to run an experiment with the vehicle to a single person. Moreover, the vehicle is small enough to transport in a conventional car. This vehicle has been conceived as an AUV, hence it carries its own source of power, which provides about an hour of autonomy. The vehicle can also be powered by an external source using an umbilical cable. This option facilitates running long-term experiments. The hull has been designed as a sphere, as illustrated in Figure 1.9. As a result, it offers equal hydrodynamic coefficients in any direction. There have been precedents with this shape, such as the ODIN AUV from the University of Hawaii (U.S.A.) or the ROBIN robot from CNRIAN (Italy), as well as the HYBALL from the Heriot-Watt University (U.K.). The hull consists of two stainless steel hemispheres joined with wing nuts and bolts.



(a) (b)

Figure 1.9. (a) Synthetic image of URIS AUV. (b) Photograph of URIS AUV.

The mass of the vehicle has been distributed in such a way that the center of mass is below the buoyancy center (as in the GARBI prototype) making the vehicle passively stable in *roll* and *pitch*. Propulsion is achieved by means of 4 thrusters placed equidistant on the exterior of the vehicle, as illustrated in Figure 1.9(a). Due to its stability in *pitch* and *roll*, there are only four degrees of freedom; *X*, *Y*, *Z* and *Yaw*. The vehicle incorporates a magnetic compass, a pressure sensor, water speed sensors, DGPS, water leakage sensors and computer vision system. An on-board Pentium PC 104 is in charge of the control of the vehicle's sub-systems. An 80552-based microcontroller is used to reduce the computing load of the main computer, taking charge of peripheral computations. During development, the optional umbilical cable is used to connect the on board computer to the surface computer where the development environment (Tornado/VxWorks) resides. The sphere radius is about 35 cm and the weight is approximately 35 kg. Table 1.3 summarizes the main characteristics of this robot.

Table 1.3. Characteristics of URIS.

Type	Autonomous Underwater Vehicle (AUV)
D.O.F.	4 (*x*,*y*,*z*,*Yaw*)
Stability	Passively stable in Roll and Pitch
Propulsion	4 thrusters (20W x 15V DC motor + dynamo)
Energy	4 packages of NiCd batteries (50 W x 12V)
Max. depth	30 meters
Sensors	Magnetic compass (*Yaw*)
	Pressure sensor (*z*)
	Vision system (RGB+laser)
	Speed sensor
	DGPS
	Water and battery charge detection

1.4 Objectives

The main objective of this work is to design and develop a vision-based sensing system to serve as a positioning tool for an underwater vehicle. This sensing system should have the capabilities of automatic sea-floor mapping and station keeping, in order to aid the vehicle to perform local navigation.

Taking into account the main goal described above, the research described in this thesis will meet the following specific objectives:

- Analyze and compare the existing techniques to create visual maps of the ocean floor, pointing out the advantages and drawbacks of the different approaches.
- Design and implement a new system to create maps of the bottom of the sea (known as *visual mosaics*). This system should be able to run in real time and provide robust estimates of the vehicle motion as the map is being constructed. This capability of simultaneous localization and map building is known as *Concurrent Mapping and Localization* (CML) in the literature [Leo92,Smi97].
- Propose a framework to update the position estimates as new information becomes available.
- Construct a testing system to evaluate the accuracy of the vision-based sensor against truth measurements.

1.5 Organization

The different chapters of this work outline a strategy for detecting the position of an underwater vehicle through the construction of a mosaic. This mosaic is, therefore, a visual map which the vehicle utilizes to navigate.

This dissertation has been structured as follows:

Chapter 2 establishes a theoretical basis for the methods and strategies presented in subsequent chapters. The basis of the different tools which are used in our approach to solve the problem of underwater mosaic construction is described. These tools range from the properties of projective geometry to the characterization of image texture, or optimal estimation techniques. Since we are working with underwater vehicles, the problems caused by the transmission properties of the medium are also considered.

Chapter 3 analyzes a review of the existing mosaicking systems for UUV navigation. The advantages and drawbacks of the presented algorithms are compared and analyzed. A comparative table illustrating the main differences of the existing methods is provided at the end of the chapter.

The fourth Chapter proposes some alternatives to solve the correspondence problem in an accurate and reliable way. The use of texture analysis as a characterization feature which improves the quality of the detected correspondences is demonstrated in this Chapter. At the end of the Chapter, a new approach to detect correspondences in consecutive images of a sequence is proposed. The partial results of this phase are also presented in this Chapter.

Chapter 5 details the phases of a methodology to construct visual mosaics of the underwater environment while simultaneously using this map to compute absolute vehicle location (CML). This methodology makes use of the strategy proposed in Chapter 4 to detect reliable correspondences, and applies a robust regression technique is used to estimate the planar transformation which explains the apparent motion of the image. Then, this planar transformation is used to relate every image to the common mosaic frame, constructing a composite image with all the frames of the sequence.

As the mosaic increases in size, image local alignment errors increase the inaccuracies associated to the position of the vehicle. For this reason, the sixth Chapter extends the mosaicking system of Chapter 5 to obtain trajectory estimates from the information of the crossover trajectories in the mosaic. In this way, when the arbitrary path of the submersible describes a loop, the images forming the mosaic are re-aligned and a better position estimation is obtained.

In Chapter 7, the different experimental results obtained from simulations, laboratory tests with real images, and sea trials are presented. A method to evaluate the accuracy of the path reconstructed from the mosaic is described.

Finally, Chapter 8 presents the conclusions derived from this work. A summary of the contributions of this dissertation is presented, and the future directions of research are outlined.

The bibliographical references have been located at the end of the corresponding Chapter.

References

[An96] P.E. An, A.J. Healey, S.M. Smith and S.E. Dunn, “New experimental results on GPS/INS navigation for Ocean Voyager II AUV”, in *Proceedings of the IEEE Symposium on*

Autonomous Underwater Vehicle Technology, pp. 249–255, 1996.

[Fos95] T.I. Fossen, "Guidance and Control of Ocean Vehicles," John Willey & Sons, 1995.

[Jor93] K.V. Jorgensen, B.L. Grose and F.A. Crandall, "Doppler sonar applied to precision underwater navigation", in *Proceedings of the MTS/IEEE Oceans*, vol. 2, pp. II469–II474, 1993.

[Lar00a] M.B. Larsen, "High performance Doppler-inertial navigation-experimental results", in *Proceedings of the MTS/IEEE Oceans*, vol. 2, pp. 1449–1456, 2000.

[Lar00b] M.B. Larsen, "Synthetic long baseline navigation of underwater vehicles", in *Proceedings of the MTS/IEEE Oceans*, vol. 3, pp. 2043–2050, 2000.

[Law98] A. Lawrence, "Modern Inertial Technology", *Springer-Verlag*, New York, 1998.

[Leo92] J. J. Leonard and H.F. Durrant-Whyte, "Directed Sonar Sensing for Mobile Robot Navigation", Kluwer Academic Publishers, 1992.

[Mar82] D. Marr, "Vision," W.H. Freeman and Company, New York, 1982.

[Rid01] P. Ridao, "A hybrid control architecture for an AUV", Ph.D. Thesis, University of Girona, 2001.

[Smi97] C.M. Smith, J.J. Leonard, A.A. Bennett and C. Shaw, "Feature-based concurrent mapping and localization for autonomous underwater vehicles", in *Proceedings of the MTS/IEEE Oceans Conference*, vol. 2, pp. 896–901, 1997.

[Tho97] H. Thomas and E. Petit, "From autonomous underwater vehicles (AUV's) to supervised underwater vehicles (SUV's)", in *Proceedings of the MTS/IEEE Oceans*, vol. 2, pp. 875–887, 1997.

[Wil60] W.D. Wilson "Speed of sound in sea water as a function of temperature, pressure and salinity, Journal of the Acoustic Society of America, vol. 32, 1960.

[You91] J.W. Younberg, "Method for extending GPS to underwater applications", U.S. Patent #5,119,341, 1991.

[Yun99] X. Yun, E.R. Bachmann, R.B. McGhee, R.H. Whalen, R.L. Roberts, R.G. Knapp, A.J. Healey and M.J. Zyda, "Testing and evaluation of an integrated GPS/INS system for small navigation", *IEEE Journal of Oceanic Engineering*, vol. 24, no. 3, pp. 396–404, 1999.

Chapter 2

Theoretical Background

This chapter establishes a theoretical basis for the methods and strategies presented in subsequent chapters. Our approach to solve the problem of underwater mosaic construction involves using tools of different nature, ranging from the properties of projective geometry to the accurate use of image texture, or optimal estimation techniques. Concretely, this chapter starts with the exposition of the general assumptions that delimit the work of this dissertation. Then, the peculiarities of the underwater environment are analyzed from a computer vision viewpoint, emphasizing the aspects that directly affect image-processing algorithms. Next, an accurate analysis of image texture is performed. Texture will allow to a large extent the improvement of the detection of feature correspondences in image pairs. Then, the general aspects of classical projective and algebraic geometry are studied in the frame of fundamental problems in computer vision. The projective geometry will explain how is it possible to align the different images forming the mosaic by means of projective transformations or collineations. Next, these basis are used to explain the modeling of the distortion produced by the camera lenses and the ray diffraction at the water-camera housing and the air-camera housing interfaces. Finally, some aspects of stochastic processes and Kalman Filtering are described. These

techniques will allow the integration of any new information regarding the vehicle positioning while the visual mosaic is being constructed.

2.1 General assumptions

In order to keep the work described in this thesis in a realistic plane, some assumptions have to be made. The algorithms detailed in the following chapters are valid only if the following assumptions are met:

- The underwater vehicle carries a camera looking down to the seabed.
- The turbidity of the water allows a reasonable visibility at the working area.
- The submersible, and therefore the camera carried by the vehicle, is close to the ocean floor. This proximity aspect is quite related to the previous point. Consequently, we understand “closeness” as a function of the visual range allowed by the underwater medium.
- The underwater terrain is assumed to be planar. This hypothesis can be relaxed in the sense that the differences in depth within the seabed are assumed to be negligible with respect to the average distance from the camera to the seabed.
- Lighting conditions are adequate to obtain images of the ocean floor.

Definitively, the considerations stated above assume that the images captured by the camera contain sufficient information to be processed by means of image processing strategies. It will be assumed for the rest of this dissertation that these assumptions are met, and when they are not fulfilled or additional assumptions are made, it will be clearly stated in the text.

2.2 Underwater optical imaging

The application of standard computer vision techniques to underwater imaging involves dealing with additional problems. This is mainly due to the transmission properties of the medium [Fun72]. The optical properties of different water bodies depend on the interaction between the light and the aquatic environment. This interaction includes basically two processes [Neg95]: absorption and scattering (see Figure 2.1).

- **Absorption.** It is the process whereby light energy is converted to a different form (principally heat). Therefore, light disappears from the image-forming process.
- **Scattering.** It is produced by change of direction of individual photons, mainly due to the different sizes of the particles forming the water. It is nearly independent of wavelength. Scattering can be further divided into backscatter and forward scattering:
  - Backscatter appears when the light is reflected in the direction of the imaging device.
  - Forward scattering is produced when the light reflected by the imaged object suffers from small changes in its direction. This effect normally produces a blurring of the object when viewed from the camera.

Backscattering is normally reduced by increasing the distance (*l*) between the light source and the imaging device, and forward scattering can be attenuated by decreasing the distance *Z* to the sea floor (or the imaged object).

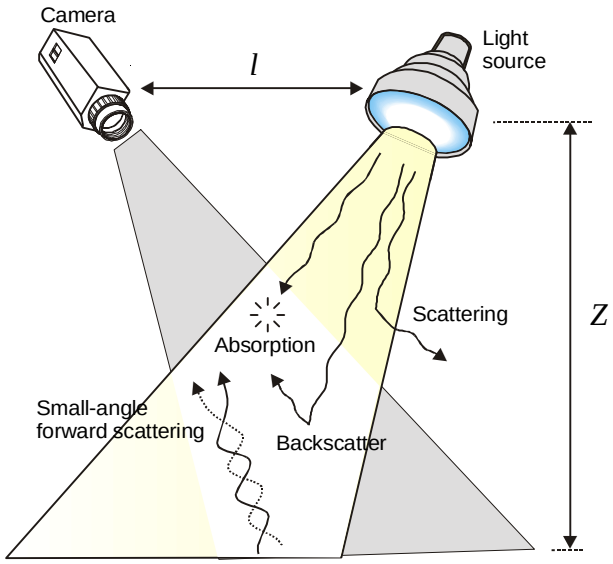


Figure 2.1. Lighting problems to be faced in underwater image processing.

For all the transmission properties of the media described above, processing of underwater images suffers from the following problems:

- **Lack of image features.** They often lack distinct features (*e.g.* points, lines or contours) that are commonly exploited in terrestrial vision systems for tracking, positioning, navigation, etc. There are two main reasons for this absence of features: firstly, the sea-floor lacks well-defined contours. Even man-made objects such as pipes or cables loose their straightness due to the proliferation of marine life. Secondly, the light reflected by the objects suffers from forward scattering [Jaf91] (as described above), which causes a blurring of these elements in the image.
- **Limited range.** The range is limited due to light absorption and the need for artificial light introduces many new properties to the image, such as low contrast and non-uniform illumination.
- **Clutter and lack of structure.** Sub-sea scenes frequently present little structure and high clutter in the regions of interest for exploration.
- **Marine snow.** Quite often, small observable particles floating in the water show up as marine snow making feature extraction difficult (backscattering). This effect is due to suspended particles, as much as the own water molecules [Car94].
- **Image distortion**. A first ray diffraction is produced at the water-camera housing, and a second one occurs at the air-camera housing interfaces, reducing the effective focal length of the camera.

However, there is generally an advantage in the processing of underwater imaging with respect to most terrestrial applications [Neg00], we are dealing with the 3D rigid body motion of the underwater vehicle relative to a nearly motionless background: the seafloor environment.

2.3 Image Texture

First of all, we should bear in mind why textures have been considered in this work. We aim to develop a mosaicking system based on feature-based matching. In fact, determining which feature in the first image corresponds to which item in the next

one is known as the *correspondence problem*. Existing methods for solving the correspondence problem normally assume that [Tru98]:

(i) Most scene points are visible from both images.
(ii) Corresponding image regions are similar.

Unfortunately, these assumptions may be false, and the correspondence problem becomes considerably difficult. Hence, establishing correspondences between features over the images is a process which is prone to gross error [Aye95], and it does not exist a completely reliable method. For this reason, we propose the use of image textures to provide more robustness to the matching process. We believe that an accurate use of textures as characterization features canimprove to a large extent the quality of the detected matches, thereby benefiting the overall mosaicking process (as will be later shown in Chapter 4).

2.3.1 Introduction

It is difficult to find a universal definition of image texture, however, the following intuitive properties of textures are admitted as essential characteristics [Tuc93]:

- Texture is a surface property, *i.e.*, a texture is not defined on a single point. Therefore texture is a property in which the obtained values are taken into account considering a spatial neighborhood.
- Image texture depends on the scale or resolution at which it is displayed. A texture with specific characteristics in a sufficiently small scale could become a uniform texture if it is displayed at a larger scale.
- A region contains a texture when we can find a large amount of primitives in that region. If only some primitives are creating the texture, they could eventually be perceived individually instead of composing a texture.

In this work, we propose the use of image textures in order to characterize certain zones of the images. In addition to standard correlation techniques, the information that can be extracted from the textures of the objects, could help us to reach an improvement in the establishment of correspondences between pairs of images.

When studying texture, three approaches can be followed: statistical, structural and frequential. In the first case, the statistical values of the gray-levels of an area or the number of peaks or valleys or any other spatial property are analyzed. In the second case, the structural approach tries to identify “structural” patterns in the image. It is based on the existence of a set of primitives which compose the texture. These primitives are combined by means of a set of placement rules. Finally, the last case takes into consideration that the texture is a spatial repetition of a pattern; therefore, high values of repetition are found in its Fourier transform at the repetition frequency.

Since our aim is to obtain real-time performance, we have discarded the use of the frequential approach, since it has a considerable computational cost. On the other hand, it has been proved that the statistical methods are very adequate to describe textures with a random gray-level distribution [Hue96]. Moreover, according to [Jah95], the gray-level distribution in natural textures has a random behavior. Therefore, the statistical approach seems more adequate than the structural technique to process underwater images.

2.3.2 Statistical Texture Analysis

Statistical texture operators analyze the distribution of a given property for every pixel of the image. Thereby, the statistical approach tries to describe the spatial distribution of the gray levels of the pixels on the image by searching rules of distribution and relationships among these levels. According to these methods, the analysis of the textures consists in computing the characteristics for every point that belongs to the image of the texture. These characteristics are defined as the combination of intensities in different positions of the image. These positions are, by no means, random: they have a specific emplacement with respect to the point which is being evaluated. This definition involves two characteristics:

- The application of a statistical measure to every single pixel of the image results in a new image where the gray level of every pixel reflects the value of the statistical measure in that point.

- The value of the statistical measure does not only depend on the current pixel which is being evaluated but also depends on a set of pixels located on given positions with respect to itself, *i.e*. its neighboring pixels.

Next, the texture operators which have been analyzed are described. Every texture operator described below has a series of parameters which define its behavior. Typical parameters range from distances, angles, to size of the neighborhood, etc., giving rise to a high number of combinations when selecting a texture operator.

First order statistics

First order statistics measure the probability of having a specific value on the point which is being evaluated. The histogram of the image will become an important tool to calculate this statistical value. Then, by looking at the image histogram *h*, the following first order properties can be computed:

$$\text{Mean:}\quad \mu = \sum_{i=1}^{n} i\, h(i) \tag{2.1}$$

$$\text{Standard deviation:}\quad \sigma^2 = \sum_{i=1}^{n} (i-\mu)^2 h(i) \tag{2.2}$$

$$\text{Third moment:}\quad \mu_3 = \frac{1}{\sigma^3} = \sum_{i=1}^{n} (i-\mu)^3 h(i) \tag{2.3}$$

$$\text{Entropy:}\quad \varepsilon = -\sum_{i=1}^{n} h(i) \log h(i) \tag{2.4}$$

Where the *mean* is the estimation of the gray-level of the texture; the *standard deviation* shows the average dispersion with respect to the mean. The *third moment* measures the asymmetry of the histogram, while the *entropy* measures the histogram uniformity.

However, since the above properties are based on the histogram, they do not perceive the spatial information. That inconvenience is solved by the texture operators described below.

Co-occurrence Matrix

This operator [Har73] searches the repeated occurrence of pairs of gray-level configurations in the image according to two parameters: the distance and the angle defined by the pair of points.

Consider *d* as the distance between two pixel positions. The immediate neighbors of any pixel can lie on four possible directions: $\theta \in \{0^o, 45^o, 90^o, 135^o\}$, as shown in Figure 2.2. The co-occurrence matrix computes the probability of two given gray-levels to appear in the image at a distance *d* and angle θ . Then, for a given *d* and θ the rows and columns of the matrix represent the different gray levels of the image, and every position of the matrix corresponds to the frequency of occurrence of that combination of intensities.

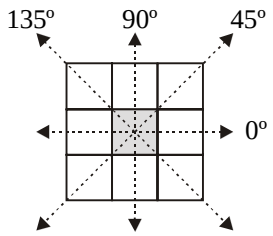


Figure 2.2. Possible directions of the neighbors of a pixel, considering distance *d*=1.

Many properties can be measured from the co-occurrence matrix. The set of statistics illustrated in Figure 2.3 is computed for every co-occurrence matrix, obtaining the textural characteristics of the image.

$$Probability = \max_{i,j=0}^{N-1} (m_{ij}) \qquad Entropy = \sum_{i,j=0}^{N-1} m_{ij} \times \log(m_{ij})$$

$$Uniformity = \sum_{i,j=0}^{N-1} m_{ij}^2 \qquad Contrast = \sum_{i,j=0}^{N-1} (i-j)^2 \times m_{ij}$$

$$Inverse = \sum_{i,j=0}^{N-1} \frac{m_{ij}}{(i-j)^2} \quad i \neq j \qquad Correlation = \sum_{i,j=0}^{N-1} \frac{(i-\mu)\times(j-\mu)\times m_{ij}}{\sigma^2}$$

$$Homogeneity = \sum_{i,j=0}^{N-1} \frac{m_{ij}}{1+|i-j|} \qquad \mu = \sum_{i,j=0}^{N-1} i \times m_{ij}$$

$$\sigma = \sqrt{\sum_{i,j=0}^{N-1} (i-\mu)^2 \times m_{ij}}$$

Figure 2.3. Statistical measures performed to characterize the texture.
m_{ij} is the element of row *i* and column *j* of the co-occurrence matrix

Energy Filters

These operators [Law80] are derived from the computation of a series of statistical measures (basically *mean* and *standard deviation*) on a pre-filtered image. This pre-filtered image is obtained by applying a set of masks (3×3 or 5×5) which define some textural properties of the image. In order to obtain these masks, a series of vectors defining some textural proprieties are combined.

Consider the 1×3 vectors *Level*, *Edge* and *Spot*, which are defined as:

```
L3 = [ 1  2  1]
E3 = [-1  0  1]
S3 = [-1  2 -1]
```

These vectors represent the one-dimensional operations of center-weighted local averaging, symmetric first differencing (edge detection) and second differencing (spot detection). By combining these vectors in pairs, the following 1×5 vectors are obtained:

```
L5 = [ 1  4  6  4  1] = L3*L3
E5 = [-1 -2  0  2  1] = L3*S3
S5 = [-1  0  2  0 -1] = S3*S3
W5 = [-1  2  0 -2  1] = L3*E3
R5 = [ 1 -4  6 -4  1] = -E3*S3
```

which correspond to the vectors *Level*, *Edge*, *Spot*, *Wave* and *Ripple*. Combining again these vectors, by multiplying the column vectors of length 3 or 5 by row vectors of the same length, a set of 3×3 and 5×5 masks can be obtained. Figures 2.4 and 2.5 show some of the considered masks.

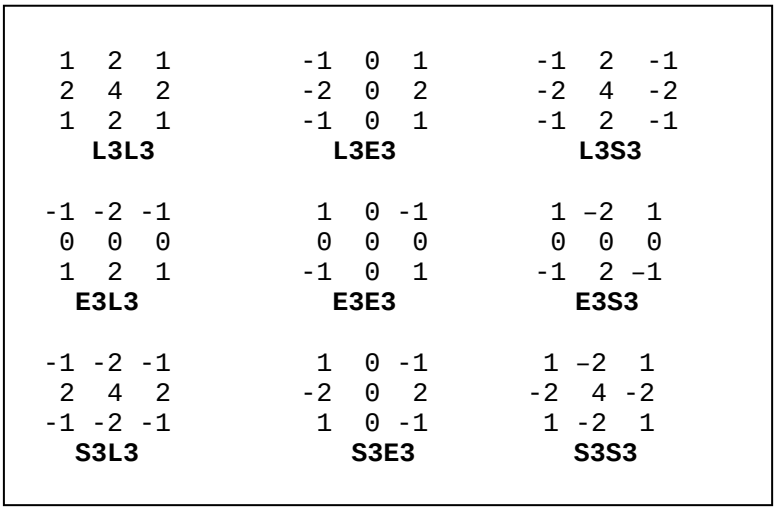


Figure 2.4. 3×3 masks applied by the energy filter

-1	0	2	0	-1
-2	0	4	0	-2
0	0	0	0	0
2	0	-4	0	2
1	0	-2	0	1

E5S5

-1	0	2	0	-1
-4	0	8	0	-4
-6	0	12	0	-6
-4	0	8	0	-4
-1	0	2	0	-1

L5S5

-1	-4	-6	-4	-1
-2	-8	-12	-8	-2
0	0	0	0	0
2	8	12	8	2
1	4	6	4	1

E5L5

Figure 2.5. Sample 5×5 masks after multiplication of the indicated vectors of size 5.
The complete set of 5×5 masks can be found in [Law82]

Once the image has been convolved with these masks, a set of statistical measures can be computed for every resulting image (mean, standard deviation and positive/negative mean):

$$\text{Absolute mean: } \mu = \frac{\sum_{i=1}^{n} |c_i|}{n} \tag{2.5}$$

$$\text{Standard deviation: } \sigma = +\sqrt{\sum_{i=1}^{n} \frac{(c_i - \mu)^2}{n}} \tag{2.6}$$

$$\text{Positive mean: } \mu^{+} = \frac{\sum_{i=1}^{n} c_i}{n}, \text{ with } c_i >= 0 \tag{2.7}$$

$$\text{Negative mean: } \mu^{-} = \frac{\sum_{i=1}^{n} c_i}{n}, \text{ with } c_i < 0 \tag{2.8}$$

These statistical measures properly indicate the energy of the texture. They are applied to a neighborhood around the point where the texture is to be evaluated. The above equations consider a region of size *n*, and c_i represents the i^{th} element of this neighborhood.

Local Binary Patterns

Local Binary Patterns (LBP) try to reduce the complexity of the previous texture operators. They are based on a model of texture analysis described in [Wan91], where a textured image is characterized by its texture spectrum. This texture spectrum is composed by the 8 neighbors of the selected pixel, every neighbor having a value $g = [0,1,2]$. The combination of values of the set of neighbors is known as *texture unit*. Ojala *et al.* introduced in [Oja96] a variant of this idea, reducing the value of every pixel to $g = [0,1]$. Then, the number of possible texture units is shorten to $2^8 = 256$, instead of $3^8 = 6561$.

LBP work as follows: for every pixel of the image, its 3×3 neighborhood is selected as region to analyze. The 8 neighbors of this region are compared to the center pixel, creating a texture unit which has a value of 1 is the neighbor is greater or equal to the center pixel, or 0 otherwise. Next, the binary values of this texture unit are multiplied by the binomial weights given to the corresponding pixels, as shown in Figure 2.6. Finally, the sum of the eight resulting values is computed, giving rise to the texture value of this region (196 in the example of Figure 2.6).

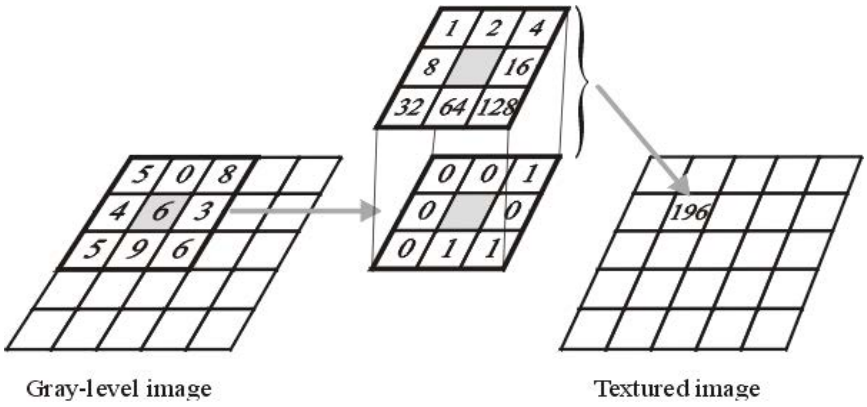


Figure 2.6. Local Binary Patterns (LBP). The pixels with a gray level higher than the center pixel are multiplied by the binomial weights given to the corresponding pixels and the obtained values are summed for the LBP number of this texture unit (4+64+128 = 196).

Contrast Operator

LBP describes the spatial structure of the local texture, but it does not address the contrast of the texture. For this purpose, Ojala *et al.* proposed [Oja99] a contrast measure. The contrast feature simply consists of performing a gray-scale differentiation in the region which is being considered. The neighboring pixels are compared with the selected point, computing the average of those neighbors with a gray-value higher than that of the center pixel. A second average is computed with the neighbors with an intensity value below the selected pixel. Then, the difference of both averages is computed. This value is known as **contrast** of the texture. Figure 2.7 summarizes the functioning of this texture operator.

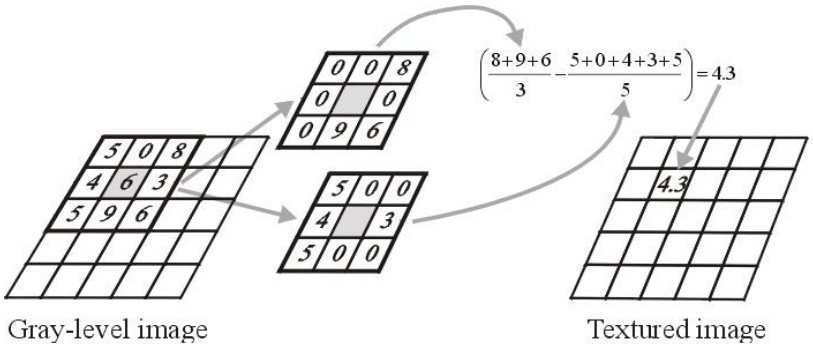


Figure 2.7. Contrast Operator

Center-symmetric covariance measure

It analyses the textures by performing some operations on the symmetric pixels surrounding the point which is being characterized [Har95]. Consider the 3×3 neighborhood of Figure 2.8, defining the center-symmetric pairs of pixels (g_i, g_i').

g_2	g_3	g_4
g_1		g_1'
g_4'	g_3'	g_2'

Figure 2.8. 3×3 neighborhood with 4 center-symmetric pairs of pixels.

By looking at these pixels, four rotation-invariant measures can be computed: two local center-symmetric auto-correlations, with linear and rank-order versions (SAC and SRAC), together with a related covariance measure (SCOV) and variance ratio (SVR).

$$\text{SAC}=\frac{\text{SCOV}}{\text{VAR}} \tag{2.9}$$

$$\text{SRAC}=1-\frac{12\left[\sum_{i=1}^{4}\left(r_i-r_i'\right)^2+\frac{1}{12}\sum_{j=1}^{l}\left(t_j^3-t_j\right)\right]}{n^6-n^2} \tag{2.10}$$

$$\text{SCOV}=\frac{1}{4}\sum_{i=1}^{4}\left(g_i-\mu\right)\left(g_i'-\mu\right) \tag{2.11}$$

$$\text{SVR}=\frac{\text{WVAR}}{\text{BVAR}} \tag{2.12}$$

where μ is the local mean, r_i is the rank of the gray-level of pixel *i*, *l* defines the number of different ranks, each t_i is the number of ties at each r_i, and *n*=3 is the size of the considered neighborhood. For further details on these equations see [Har95].

The local variance VAR (=BVAR+WVAR) is a measure of local gray-scale variation, very sensitive to noise and other local gray-scale transformations, as well as the between-pair variance (BVAR) and the within-pair variance (WVAR). These variances are defined as follows.

$$\text{VAR}=\frac{1}{8}\sum_{i=1}^{4}\left(g_i^2+g_i'^2\right)-\mu^2 \tag{2.13}$$

$$\mathrm{BVAR} = \frac{1}{16}\sum_{i=1}^{4}\left(g_i - g_i'\right)^2 - \mu^2 \tag{2.14}$$

$$\mathrm{WVAR} = \frac{1}{16}\sum_{i=1}^{4}\left(g_i - g_i'\right)^2 \tag{2.15}$$

All the above texture operators are abstract measures of texture pattern and gray-scale, providing discrimination information about the amount of local texture.

2.4 Projective Geometry

This section introduces the concepts of projective geometry in general and planar geometry in particular. The geometry of projective transformations of the plane models the geometric distortion which arises when a plane is imaged by a perspective camera. Projective geometry models this imaging and also provides an adequate mathematical representation for computations. In the context of computer vision, the use of projective geometry allows the expression of some geometrical relationships in terms of linear equations, which makes them easier to deal with.

2.4.1 Definition of Projective Space

The set of points represented by a n+1 vector $\left(x_1, x_2, \ldots, x_n, x_{n+1}\right)^{\mathrm{T}} \in \mathbb{R}^{n+1}$ of real values is called a **Projective Space** $\mathbb{P}^n$ if and only if at least one of the n+1 vector coordinates is different from zero and two vectors $\left(x_1, \ldots, x_n, x_{n+1}\right)^{\mathrm{T}}$ and $\left(\ddot{e}x_1, \ldots, \ddot{e}x_n, \ddot{e}x_{n+1}\right)^{\mathrm{T}}$ represent the same point for any $\ddot{e} \neq 0$. The elements of a projective space vector are usually called *homogeneous* or *projective coordinates* [Fau93].

2.4.2 The 2D projective plane

In general, we can say that a linear transformation of a projective space $\mathbb{P}^n$ is defined by a non-singular $(n+1)\times(n+1)$ matrix **A**. This transformation is also known as *collineation* or *projective transformation*. The matrix **A** performs an invertible mapping of $\mathbb{P}^n$ onto itself, and is defined up to a non-zero scale factor. In the case of

2D projective transformations, $\mathbb{P}^2$ is known as the *projective plane*. Its importance in Computer Vision is derived from the fact that it is useful to model the image plane as a projective plane, describing the geometrical relationships among image features. Two different views of the same planar scene in 3D space are related by a collineation in $\mathbb{P}^2$. This collineation is also known as *homography* [Sze94].

Given a point $(x, y)^{\mathrm{T}}$ in $\mathbb{R}^2$, it can be represented in homogeneous coordinates as a 3-vector by adding a final coordinate of 1: $(x, y, 1)^{\mathrm{T}}$. Therefore, an arbitrary homogeneous vector representative of a point and given by $(x_1, x_2, x_3)^{\mathrm{T}}$ represents the point $(x_1/x_3, x_2/x_3)^{\mathrm{T}}$ in $\mathbb{R}^2$, defined up to a scale factor. All the 2D points represented by homogeneous vectors belong to the projective space $\mathbb{P}^2$.

A 2D projective transformation is a linear transformation on homogeneous 3-vectors represented by a non-singular 3×3 *homography* matrix:

$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \text{ or } \mathbf{x}' = \mathbf{H}\mathbf{x} \tag{2.16}$$

where $\mathbf{x}' = (x_1', x_2', x_3')^{\mathrm{T}}$ and $\mathbf{x} = (x_1, x_2, x_3)^{\mathrm{T}}$ are the homogeneous vector representations of two points, and **H** is a matrix defining the linear mapping of homogeneous coordinates. It should be noted that matrix **H** may be multiplied by any arbitrary scale factor $\ddot{e} \neq 0$ without altering the projective transformation. Therefore, we could say that **H** is a *homogeneous* matrix, since only the ratio of the matrix elements is significant, as in the case of homogeneous points. It can be seen that there are eight independent ratios amongst the nine elements of **H**. For this reason a projective transformation has eight degrees of freedom. It accounts for the perspective mapping of a planar scene into the image plane of a camera, leaving all the projective properties of the scene invariant. In the same way, the projective transformation can relate two different images of the same planar scene, as shown in Figure 2.9. Taking into account what we have described above, Equation (2.16) can be expressed as:

$$\begin{bmatrix} \lambda\, x' \\ \lambda\, y' \\ \lambda \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.17}$$

where (x', y') and (x, y) are the image coordinates of the same 3D point, viewed from images I' and I, respectively; and $\ddot{e} \neq 0$ is a scaling factor. This inter-image projective transformation can be computed by means of a minimum of four pairs of correspondences between the two images.

It should be taken into account that if the scene is static, but not planar, parallax effects would produce image misalignments, except for the case where the cameras have the same optical axis. Consequently, a projective transformation is only valid in 3D scenes if the camera translates along the optical axis or rotates around the optical axis, but no other translations or rotations are allowed. This is reasonable since we are dealing with planar transformations, which are not able to cope with 3D structures.

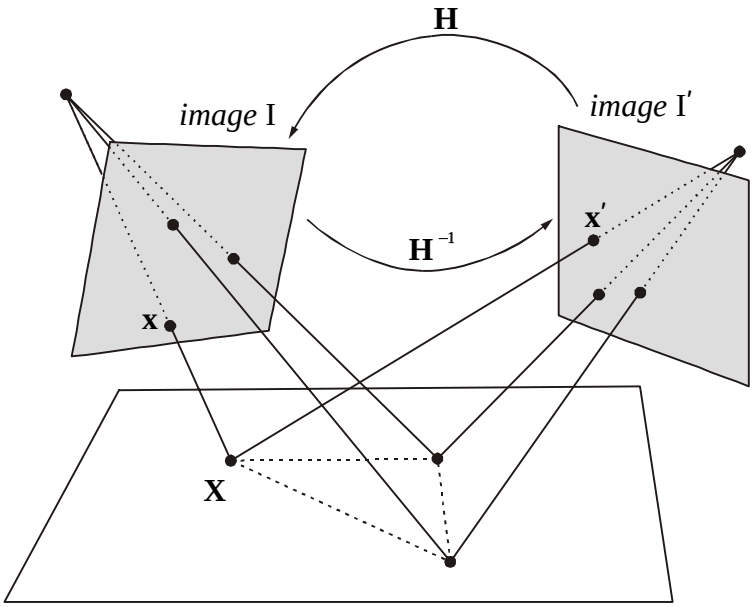


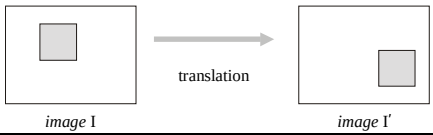
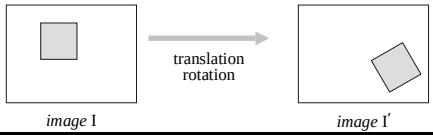
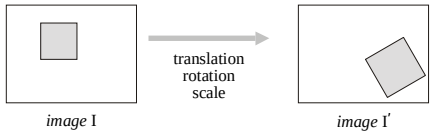
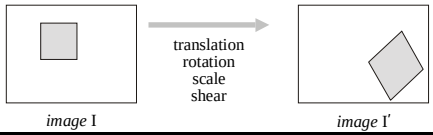
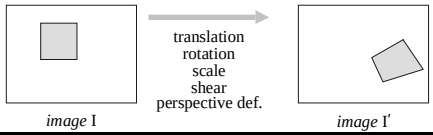
Figure 2.9. Projective transformations in perspective images. It relates two different images of the same planar scene. Note the different distribution of projected points in both images.

2.4.3 A hierarchy of homographies

A general projective transformation takes into account translation, rotation, scaling, shear and perspective deformation. Obviously, this homography may contain

more free parameters than necessary. For instance, consider a setup where the image plane is always parallel to the planar scene, the camera is not allowed to rotate around any axis, and it keeps constant its distance to the scene[1]. In this case, only the two degrees of freedom (DOF) describing translation are necessary. Table 2.1 shows the image of a square and the resulting transformations for different motion models. The first row of the table shows the case described above, where only 2 DOF are necessary. If we add rotation to the previous collineation, the Euclidean transformation is obtained. The similarity and affine transformations are obtained if when adding a scaling factor and shear, respectively. Finally, the last row of the table shows the most general motion model, which also includes perspective deformation.

Table 2.1. Motion models to describe a planar transformation. DOF: degrees of freedom.

Model	Homography Matrix	Distortion
Pure translation 2 DOF	$\begin{bmatrix} \lambda x' \\ \lambda y' \\ \lambda \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$	
Euclidean 3 DOF	$\begin{bmatrix} \lambda x' \\ \lambda y' \\ \lambda \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$	
Similarity 4 DOF	$\begin{bmatrix} \lambda x' \\ \lambda y' \\ \lambda \end{bmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$	
Affine 6 DOF	$\begin{bmatrix} \lambda x' \\ \lambda y' \\ \lambda \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$	
Projective 8 DOF	$\begin{bmatrix} \lambda x' \\ \lambda y' \\ \lambda \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$	

In fact, when applying the 2D transformation matrices of Table 2.1, only in the projective homography the scaling factor ë would be different from 0, since all the

[1] This is the case of mosaicking with some electronic microscopes or wafer masking systems for integrated circuits.

matrices have been scaled by imposing $h_{33} = 1$. It should be noted that not always the most generic transformation will provide the best results in the determination of the apparent motion between two images. Consider again the example of the microscope sequence with 2 DOF. Suppose that the imaged object is the square of Table 2.1, and the image measurements are the points located at the corners of the square. The detection of points is performed in two images: I and I', obtaining a set of four points $\mathbf{x}_i$ and the corresponding points $\mathbf{x}'_i$. In practice, the detection of points in both images is subject to a zero-mean Gaussian noise. Imagine, for the sake of simplicity, that error is produced only in the second image I', with points in the first measured perfectly. This implies that the distance between any two corners of the square $\mathbf{x}_i$ in the first image remains a constant value *d*. Then, the error in the determination of the corners of the square in image I' would entail the distance between any two $\mathbf{x}'_i$ to be different from *d*. Therefore, when computing the apparent motion by using the most general projective motion model, not only a 2D translation would be found, but many other DOF would be estimated. However, the most simple homography tries to explain the motion as only a 2D translation. This second approach would achieve a more accurate result since the knowledge about the nature of has been used to choose a motion model. Unfortunately, it is not always possible to know beforehand which is the motion model which best describes the motion of the camera.

2.4.4 Non-linear planar transformations: the case of lens distortion

The homography matrix **H** described in the previous section is only able to describe linear transformations, while the image-forming process suffers from a non-linear distortion. For this reason a further analysis of the assumed camera model is required. Moreover, one of the goals of this work is to perform measurements from images taken from a standard video camera. Therefore, a qualitative model of the camera would allow a comprehensive approximation of the camera projection. The most commonly used model for a camera is the so-called *pinhole* model [Ito91]. It assumes that each point of the scene is projected onto the plane where the image is formed by a straight line through the projection center. This plane is known as *image plane*, the projection center is known as *focal point* or *optical center* (denoted *C*) and the distance between the image plane and *C* is known as *focal length* (*f*). Figure 2.10 illustrates the pinhole camera model, considering that the optical center *C* is located

in front of the image plane. This situation is known as backprojection, since the image is formed behind the optical center. In this case the image is inverted with respect to the scene. A simpler situation can be found in Figure 2.11, where the focal point is placed behind the image plane. Both camera configurations are modeled by the same parameters, but considering a different sign of the focal length (*f*). From both of these figures it can be seen that the relationship between image coordinates (x_i, y_i) and 3D space coordinates (X_i, Y_i, Z_i) can be written as:

$$x_i = \frac{f}{Z_i} X_i \tag{2.18}$$

$$y_i = \frac{f}{Z_i} Y_i \tag{2.19}$$

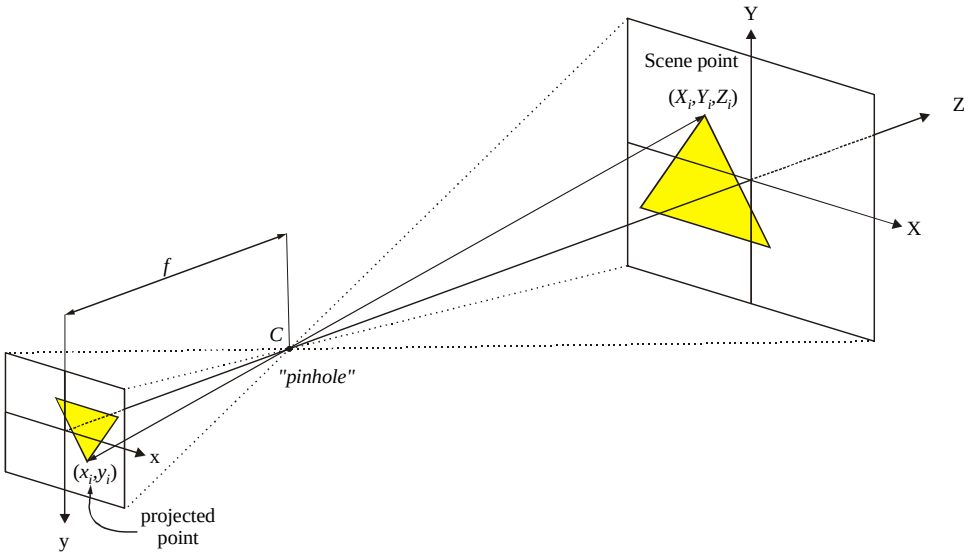


Figure 2.10. Image formation in a camera assuming a pinhole model. Backprojection model.

Equations (2.18-19) can be expressed in matricial way,

$$\begin{bmatrix} \lambda x_i \\ \lambda y_i \\ \lambda \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} \tag{2.20}$$

Unfortunately, in a practical situation the linear relationship illustrated in Equation (2.20) does not hold true. This is due to some types of imperfections in the design and assembly of the lens composing the optical system, which distorts the projection of points in the image plane. In the case of underwater imaging, the non-

linear distortion comes from two sources: the first one is the imperfection of the camera lens, and the second comes from the light beam deflections at the water-camera and air-camera housing interfaces.

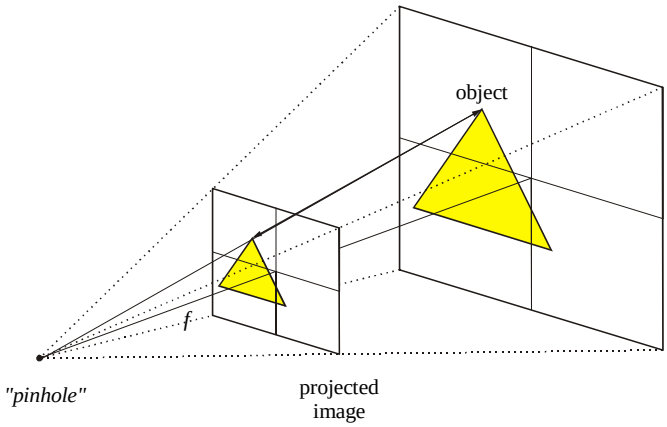


Figure 2.11. Configuration of the pinhole camera model where the image is placed in front of the focal point (*frontprojection*).

The total distortion can be divided into two components: radial component and tangential component [Sla80].

Radial distortion: The radial distortion causes an inward or outward displacement of a given image point from its ideal projection. This type of distortion is mainly caused by flawed radial curvature of the lens. A negative radial displacement of the image points is referred to as barrel distortion. It causes outer points to spread and the scale to increase. This displacement can be modeled by the following equations:

$$\begin{bmatrix} x_u \\ y_u \end{bmatrix} = \begin{bmatrix} x_d + x_d\left(k_1 r^2 + k_2 r^4 + ...\right) \\ y_d + y_d\left(k_1 r^2 + k_2 r^4 + ...\right) \end{bmatrix} \tag{2.21}$$

where $\left(x_d, y_d\right)$ are the distorted (measured) coordinates of the point, and its undistorted coordinates are $\left(x_u, y_u\right)$; $r = \sqrt{x_d^2 + y_d^2}$ is the radial distance from the projection of the focal point on the image plane (known as *principal point*), and $k_1, k_2, ...$ are the coefficients of the radial distortion.

Tangential distortion: The optical centers of the lens elements are not strictly collinear. This produces various degrees of decentering in the projection of

points on the image plane. This effect is know as tangential distortion, and can be modeled by another infinite series. However, since most of the distortion is produced by the radial component, in many cases the tangential distortion is not taken into account.

Once the image distortion has been removed, it is possible to perform metric computations from the pixels measured on the image. This is possible only if the parameters which model the internal geometry of the camera are known (provided the optical characteristics are already known). Some of these *internal* camera parameters have already appeared in the text:

- Focal length (f): it measures the distance (in mm.) from the optical center to the image plane
- Conversion parameters k_x and k_y: provide horizontal and vertical adjustments to go from millimeters to pixels. They account for the number of pixels per unit of metric distance (mm.).
- Principal point of the image (x_0, y_0): it is defined by the projection of the optical center onto the image plane. It is expressed in pixels.

Therefore, a complete modeling of the camera geometry implies the estimation of five internal parameters (f, k_x, k_y, x_0, y_0), plus two or three lens distortion components (r, k_1, k_2, ...). A description about how to compute these parameters will be given in Chapter 5.

2.5 The Kalman Filter

The Kalman Filter (KF) is a recursive estimator for estimating the state of a linear system [Kal60]. It consists of set of mathematical equations that provide an efficient computational (recursive) solution of the least-squares method. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown. It is very useful in the context of this work since it provides the basis for re-aligning the mosaic map when the vehicle path crosses itself. The great popularity and widespread use of the KF has resulted in a substantial corpus of literature regarding the derivation, implementation and properties of the filter. KF equations

are stated below in this section and will be invoked in following chapters without proof. For a more thorough derivation of the Kalman Filter and detailed discussion reference should be made to [Kal60, Kal61, May82].

The alignment of images within the mosaic is subject to the propagation of errors. As the vehicle moves, uncertainty about its position increases. Therefore, if the vehicle revisits an already mosaicked area, position uncertainty can be reduced to that the vehicle had the first time this area was explored. This is a very challenging idea since it allows the correction of error propagation in the alignment of images within the mosaic. In this way, a Kalman Filter appears as an adequate tool to improve the internal consistency of the mosaic, and at the same time it provides better estimates about the position of the vehicle.

The KF addresses the general problem of trying to estimate the state $\mathbf{x}$ of a discrete-time process that is governed by the linear stochastic differential equation [Kal60]:

$$\mathbf{x}_{k+1} = \Phi\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_{k+1} + \mathbf{w}_{k+1} \tag{2.22}$$

with a measurement $\mathbf{z}$, which corresponds to:

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \tag{2.23}$$

The $n\times n$ matrix Φ in Equation (2.22) relates the state at time step k to the state at step k+1, in the absence of process noise $\mathbf{w}_{k+1}$ and system input $\mathbf{u}_{k+1}$ (control injected at time k+1).. The $m\times n$ matrix $\mathbf{H}$ in the measurement Equation (2.23) relates the state to the measurement $\mathbf{z}_k$.

On the other hand, the random variables $\mathbf{w}$ and $\mathbf{v}$ represent the *process* and *measurement* noise, respectively. They are assumed to be independent of each other, white, and with normal probability distributions with zero mean and known covariances:

$$\mathbf{w}_k \sim N(0, \mathbf{Q}_k) \tag{2.24}$$

$$\mathbf{v}_k \sim N(0, \mathbf{R}_k) \tag{2.25}$$

The matrices describing the *process noise covariance* $\mathbf{Q}_k$ and *measurement noise covariance* $\mathbf{R}_k$ may change with each time step or measurement.

The filter is able to compute an “a priori” state estimate $\hat{\mathbf{x}}_{k+1}^{-}$ for the next time step, before a measurement $\mathbf{z}$ for time step k+1 is available.

$$\hat{\mathbf{x}}_{k+1}^{-} = \Phi_k \hat{\mathbf{x}}_k + \mathbf{B}_k \mathbf{u}_k \tag{2.26}$$

This state estimate has an "a priory" error covariance matrix $\mathbf{P}_{k+1}^{-}$:

$$\mathbf{P}_{k+1}^{-} = \Phi_k \mathbf{P}_k \Phi_k^{\mathrm{T}} + \mathbf{B}_k \mathbf{Q}_k \mathbf{B}_k^{\mathrm{T}} \tag{2.27}$$

The "super minus" $(\hat{\mathbf{x}}_k^{-})$ indicates that we are *a priori* estimating the state at step $(k+1)$ provided that the process is only known up to step *k*. Equations (2.26) and (2.27) propagate the state estimation and its covariance matrix forward in time. They are known as **propagate** equations, as shown in Figure 2.12.

Then, a gain matrix $\mathbf{K}_k$ relates the amount of influence the error between the measurement $\mathbf{z}_k$ and the previous (a priory) estimation $\mathbf{x}_k^{-}$. So, it measures how match the filter relies on the current observations $\mathbf{z}_k$, through the **update** equations:

$$\mathbf{K}_k = \mathbf{P}_k^{-} \mathbf{H}_k^{\mathrm{T}} (\mathbf{H}_k \mathbf{P}_k^{-} \mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1} \tag{2.28}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^{-} + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^{-})^{-1} \tag{2.29}$$

It can be seen from equation (2.28) that the gain matrix $\mathbf{K}_k$ is computed by using the predicted error of those elements in the state parameter vector that are obtained from the image. $\mathbf{H}_k$ gives the noiseless connections between the measurement and the state vector at time step *k*, and $\mathbf{R}_k$ is the measurement error. Then, equation (2.29) updates the estimate with the measurement $\mathbf{z}_k$. The difference between the estimated and measured parameters can be considered to be a prediction error, which is caused by either faulty measurement, faulty prediction, or a combination of both. A proportion of the predicted error is added to the parameter estimate $\hat{\mathbf{x}}_k^{-}$ to produce an updated state parameter vector $\hat{\mathbf{x}}_k$, depending on the values of $\mathbf{K}_k$.

Next, updated state error covariance $\mathbf{P}_k$ is computed:

$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right) \mathbf{P}_k^{-} \tag{2.30}$$

where the portion of the gain matrix that is associated with elements measured from the image is substracted from an identity matrix to yield a proportion of non-gain. This is multiplied with the estimated error covariance to produce an updated error covariance, reflecting the remaining uncertainty about the state parameters. This implies that as $\mathbf{K}_k$ decreases, the estimated error used to update $\mathbf{P}_k$ increases proportionally. The complete scheme of the equations involved in the filter are shown in Figure 2.12.

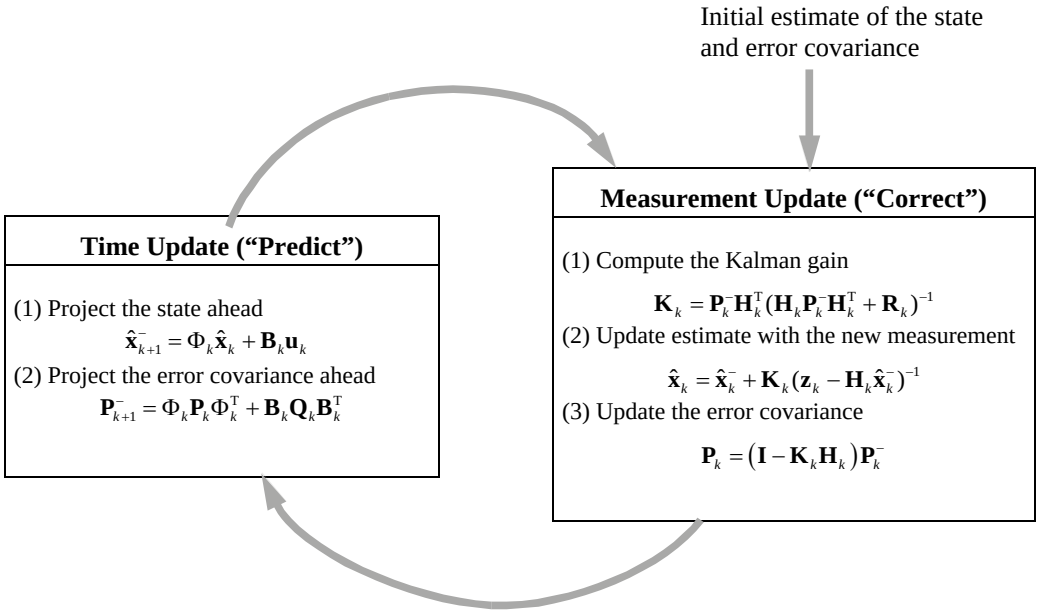


Figure 2.12. A complete picture of the operation of the Kalman filter.

Therefore, the filter estimates the state of the system at some time and then obtains feedback in the form of noisy measurements. As such, the equations for the Kalman filter perform two distinct operations: *time update* (propagation) and *measurement update* (correction). The time update equations are responsible for projecting forward in time the current state and error covariance estimates to obtain the *a priori* estimates for the next time step. The measurement update equations are responsible for the feedback, *i.e.*, for incorporating a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate. It should be noted that the time update equations could also be viewed as predictor equations, while the measurement update equations can be thought of as corrector equations.

References

[Aye95] S. Ayer, "Sequential and competitive methods for estimation of multiple motions," Ph.D. Thesis, École Polytechnique Fédérale de Lausanne, 1995.

[Car94] K.L. Carder and D.K. Costello, "Optical effects of large particles", in R.W, Spinrad, K. L. Carder, and M.J. Perry (eds.). *Ocean Optics*, Oxford monographs on geology and geophysics, chapter 13, pp. 243–257, Oxford University Press, 1994.

[Fau93] O. Faugeras, “Three-dimensional Computer Vision: A Geometric Viewpoint,” The MIT Press, Cambridge, Massachusetts, 1993.

[Fau95] O. Faugeras, “Stratification of three-dimensional vision: projective, affine and metric representations,” Journal of the Optical Society of America A, vol. 12, no.3, pp. 465–484, 1995.

[Fun72] C. J. Funk, S.B. Bryant, P.J. Beckman Jr., “Handbook of underwater imaging system design,” Ocean Technology Department, Naval Undersea Center, 1972.

[Har00] R. Hartley and A. Zisserman, “Multiple View Geometry in Computer Vision,” Cambridge University Press, 2000.

[Har73] R.M. Haralick, K. Shanmugan and I. Dinstein. “Textural features for image classification,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 3, no. 6 pp. 610–621, 1973.

[Har95] D. Harwood, T. Ojala, M. Pietikäinen, S. Kelman and L. Davis, “Texture Classification by center-symmetric auto-correlation, using Kullback discrimination of distributions,” *Pattern Recognition Letters*, vol. 16, pp. 1–10, 1995.

[Hue96] F. Huet and J. Mattioli, “A textural analysis by mathematical morphology transformations: structural opening and top-hat,” in *Proceedings of the IEEE International Conference on Image Processing*, 1996.

[Ito91] M. Ito, “Robot Vision Modeling –Camera modeling and camera calibration,” International Journal on Advanced Robotics, vol. 5, no. 3, pp. 321–335, 1991.

[Jaf91] J. S. Jaffe, “Sensors for underwater robotic vision: status and prospects,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2759–2766, 1991.

[Jah95] B. Jahne, “Digital Image Processing. Concepts, algorithms and scientific applications,” Springer Verlag 3th edition, 1995.

[Kal60] R.E. Kalman, “A New Approach to Linear Filtering and Prediction Problems,” *Transactions of the ASME - Journal of Basic Engineering*, pp. 35–45, 1960.

[Kal61] R.E. Kalman and R.S. Bucy, “New results in linear filtering and prediction theory”, *Transactions of the ASME Journal of Basic Engineering*, pp. 95-108, 1961.

[Law80] K.I. Laws, “Textured Image Segmentation,” Ph.D. Thesis, Processing Institute, University of Southern California, Los Angeles, 1980.

[May82] P. Maybeck, “Stochastic Models, Estimation and Control,” vol. 1, Academic Press, 1982.

[Neg00] S. Negahdaripour and A. Khamene, “Motion-Based Compression of Underwater Video Imagery for the Operations of Unmanned Submersible Vehicles,” *Computer Vision and Image Understanding*, vol. 79, no. 1, pp. 162–183, 2000.

[Neg95] S. Negahdaripour and C.H. Yu, “On shape and range recovery from image shading for underwater applications,” in J. Yuh (ed.), *Underwater Robotic Vehicles: design and control*, chapter 8, pp. 221–250, TSI Press, 1995.

[Oja96] T. Ojala, M. Pietikäinen and D. Harwood, “A comparative Study of Texture Measures with Classification Based on Feature Distribution,” *Pattern Recognition*, vol. 29, pp. 51–59, 1996.

[Oja99] T. Ojala and M. Pietikäinen, “Unsupervised texture segmentation using feature distributions,” *Pattern Recognition*, vol. 32, pp. 477–486, 1999.

[Sla80] C.C. Slama (ed.), “Manual of Photogrammetry,” 4th ed., *American Society of Photogrammetry*, Falls Church, Virginia, 1980.

[Sze94] Szeliski, R., “Image mosaicing for tele-reality applications,” in *Proceedings of the IEEE Workshop on Applications of Computer Vision*), pp. 44–53, Sarasota, Florida, December 1994.

[Tho97] H. Thomas and E. Petit, “From autonomous underwater vehicles (AUV’s) to supervised underwater vehicles (SUV’s),” in *Proceedings of the MTS/IEEE Oceans*, vol. 2, pp. 875–887, 1997.

[Tru98] E. Trucco and A. Verri, “Introductory techniques for 3-D computer vision,” Prentice-Hall, New Jersey, 1998.

[Tuc93] M. Tuceryan and A. Jain, “Texture Analysis,” World Scientific Publishing Co., 1993.

[Wan91] H. Wan, “Textural Filters Based on the Texture Spectrum,” *Pattern Recognition*, vol 24, pp. 1187–1195, 1991.

[Wer58] M. Wertheimer, “Principles of Perceptual Organizations,” Princenton N.J., 1958.

Chapter 3

Review of underwater vision systems for sea-bed mosaicking

This chapter surveys the alternative methods of constructing underwater mosaics that have been described in the literature. It pays special attention to analyze the advantages and drawbacks of the reviewed systems. A comparative table is provided at the end of the chapter.

3.1 Introduction

Ocean floor mosaics usually obtain the individual images to form the mosaic by setting a camera on an Unmanned Underwater Vehicle (ROV or AUV). The camera is attached to the submersible, looking down to the seabed. The acquired images cover a small area of the ocean floor, as shown in Figure 3.1.

Several alternatives have been proposed to solve the mosaicking problem [Gar00]. First, we review the different strategies. Then, the common denominator among them is used to propose a general approach to mosaic construction. This allows a comparative study. Figure 3.2 shows a set of block diagrams that broadly describe

the main different approaches that can be found in the literature regarding underwater visual-based mosaicking systems.

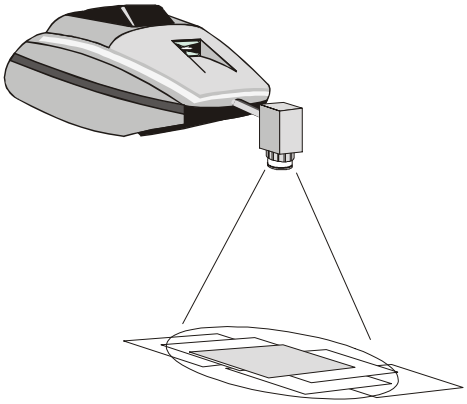


Figure 3.1. Set-up of a mosaicking system for underwater sea-floor applications

One of the first computer-aided systems to automate the construction of underwater mosaics was the one presented by Haywood in [Hay86]. In this work, mosaicking was accomplished by snapping images at well-known positional coordinates. These images were then warped together as the registration between images was known beforehand.

Some years later, IFREMER researchers set the starting point for estimating the motion of an underwater vehicle from a sequence of video images [Agu88,Agu90]. Their approach consisted of detecting and matching feature points in successive images. A Kalman Filter was used to predict the position of the features in the next image. The trajectory of the vehicle was estimated through the Generalized Hough Transform (GHT).

Fiala and Basu [Fia96] patched images together into a large composite image, in order to obtain a 3D representation of underwater objects. This vision system was used in conjunction with a ROV equipped with 3D position and orientation measuring devices. The authors limited their experiments to the mapping of planar textures onto a model of a marine vessel. No attempt was made in [Fia96] to explain how they solved the image registration problem.

Stanford University, together with the Monterrey Bay Aquarium Research Institute (MBARI), jointly developed a well-known underwater mosaicking system [Mar95]. Their system created real-time mosaics from the images provided by the OTTER semi-autonomous underwater robot, and the Ventana Remotely Operated Vehicle. This high performance was possible due to the use of special purpose hardware for image filtering and correlation. Figure 3.2(a) shows a block diagram (at the highest level of abstraction) of their dataflow. It should be noted that their system only adds a new image to the mosaic if sufficient motion is detected between the present image and the last one added to the mosaic (fixed distance intervals).

Gracias and Santos-Victor have proposed a quite accurate mosaicking strategy, based on the detection of corner points in one image and their corresponding points in the next image [Gra98b,Gra00]. The accuracy of the system is improved due to the implementation of robust outlier-detection techniques which eliminate false matches. Finally, a planar transformation matrix relates the coordinates of the two consecutive images (see Figure 3.2e). The same authors presented an alternative approach that proposes the correlation of the present image directly with the mosaic image. This improves accuracy because small errors in the estimation of motion in consecutive images do not tend to accumulate.

Researchers at the University of Miami have implemented a mosaicking system with real-time capabilities which is based on the Direct Motion Estimation algorithm [Neg98c]. This algorithm allows the estimation of the vehicle motion without the intermediate computation of image features, thus reducing the sources of error and allowing a faster computation (real-time performance without special purpose hardware). Their system only adds a new image to the mosaic every L image of the sequence (fixed time interval). To refine motion estimation, an image is extracted from the mosaic at the predicted location of the vehicle. The current actual image is then compared with the predicted image (see Figure 3.2d).

Another interesting approach is that presented in [Rzh00], where the mosaic is built based on image analysis of the frequency domain (Figure 3.2b). Their system pays special attention to the equalization of non-uniform illumination in the images to improve the Fourier-based image registration phase.

Researchers of the Woods-Hole Oceanographic Institution (WHOI) have used image mosaics to obtain a better perspective in the study of the underwater

environment [Sin98] (see Figure 3.2f). This work concentrates on the visual aspect of the mosaics, equalizing the images to eliminate differences in intensity, especially at image boundaries. They have recently proposed a featureless image registration algorithm to automatically construct visual mosaics of the ocean floor [Eus00]. The algorithm explicitly takes into account shadow effects caused by the motion of the light sources.

From those systems described above, we have chosen a representative subset of mosaicking alternatives in order to get an overview of the different philosophies that can be used. For the sake of space not all the alternatives are illustrated in Figure 3.2. Moreover, we should bear in mind that Figure 3.2 represents a quite simplified model of every mosaicking system. If we analyze the main differences among the systems illustrated above, we can realize that most of the techniques consist on comparing the present image with the previous one, or with an image extracted from the mosaic. The approach to perform this comparison significantly varies from one approach to another.

3.2 A common frame to construct mosaics

The basis of a mosaic is the computation of the displacement of the camera relative to its environment, for example, the sea floor. In order to construct a map of the ocean floor, several short-range images have to be warped together. Normally, the fusion of these images goes through some (or sometimes all) of the following steps:

1. Correction of geometric deformations (mainly due to lens distortion).
2. Lighting inhomogeneities/artifacts removal.
3. Motion detection between consecutive images of the sequence (Image registration).
4. Mosaic actualization and motion detection between the mosaic and the current frame (if the mosaic is to be actualized).
5. Image warping and mosaic construction.

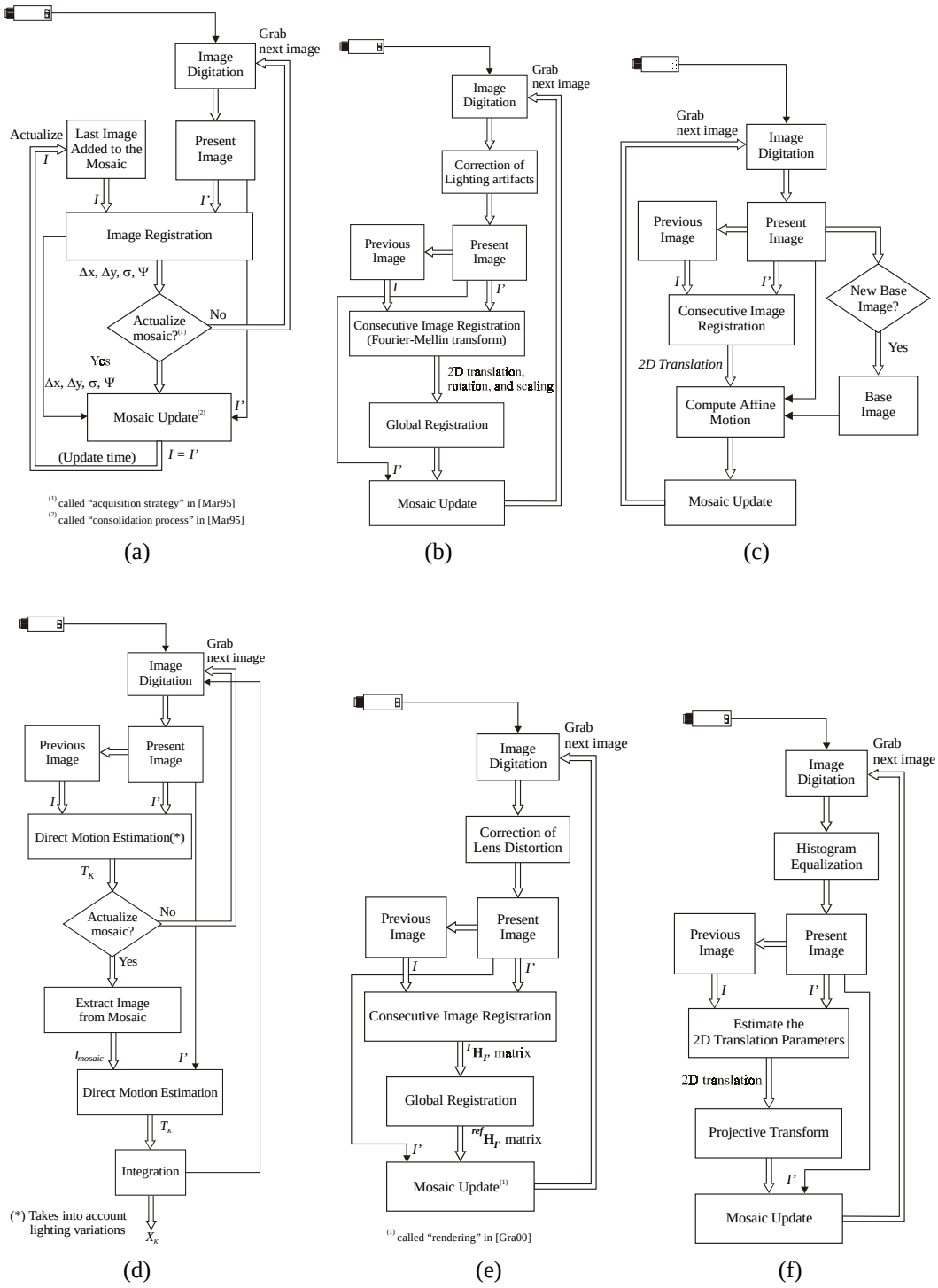


Figure 3.2. Block diagram of the different mosaicking systems proposed in the literature (a) MBARI/Stanford approach, (b) Univ. New Hampshire/Heriot-Watt, (c) Heriot-Watt Univ. (d) Univ. Miami, (e) IST&ISR, (f) WHOI

3.2.1 Correction of geometric deformations

Real-world applications cannot rely on an ideal distortion-free image. If we attempt to construct a mosaic from a sequence of well-known positions of the camera, the visual appearance of the resulting mosaic might not be satisfactory due to the discrepancies between the geometrical model of the camera and the behavior of the physical sensor itself (see Figure 3.3). A commonly used geometrical model is the *pinhole* model [Tsa87,Aya91], which assumes that the light beams pass trough a small point (pinhole), forming the image on a plane placed at a fixed distance *f* of the pinhole. This perspective projection provides a linear relationship relating a 3D-point **P** of the scene with its corresponding 2D-point **p** on the image plane:

$$\begin{bmatrix} k \cdot {}^{c}x_u \\ k \cdot {}^{c}y_u \\ k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} \end{bmatrix} \begin{bmatrix} {}^{w}X \\ {}^{w}Y \\ {}^{w}Z \end{bmatrix} \tag{3.1}$$

where $({}^{w}X, {}^{w}Y, {}^{w}Z)^T$ are the world coordinates of the 3D point **P**, $({}^{c}x_u, {}^{c}y_u)$ correspond to its perspective projection expressed in the camera coordinate system, and *k* is the scaling factor when expressed in homogeneous coordinates, obtaining $(k \cdot {}^{c}x_u, k \cdot {}^{c}y_u, k)^T$. Unfortunately, as a result of some imperfections in the design and assembly of the lens composing the optical system, the linear relationship of equation (3.1) does not hold true [Wen92]. In this way, the physical lenses introduce a non-linear distortion in the observed image points. Moreover, when treating underwater images, the ray deflections at the water-camera housing and the air-camera housing interfaces introduce a second distortion [Xu00], which normally attenuates the lens distortion. The total distortion can be modeled by a radial and tangential approximation. Since the radial component causes most of the distortion, most of the works correct only this one [Gra98a, Gar01]. Complete camera calibration is not necessary for eliminating the distortion.

Severe lens distortion can be corrected by applying several calibration algorithms [Fau86, Tsa87, Wen92]. A generic equation to compute the radial distortion is given by:

$${}^{c}x_u = {}^{c}x_d + k_1 {}^{c}x_d \left({}^{c}x_d^2 + {}^{c}y_d^2 \right) \tag{3.2a}$$

$${}^{c}y_u = {}^{c}y_d + k_1 {}^{c}y_d \left({}^{c}x_d^2 + {}^{c}y_d^2 \right) \tag{3.2b}$$

where $({}^{c}x_u, {}^{c}y_u)$ are the ideal undistorted coordinates of the measured distorted point $({}^{c}x_d, {}^{c}y_d)$, referred to the camera coordinate system and k_1 is the first term of the radial correction series.

By applying the method of Faugeras [Fau86], ${}^{c}x_d$ and ${}^{c}y_d$ in equation (3.2) should be set to:

$$ {}^{c}x_d = \frac{x_0 - {}^{i}x_d}{k_x} \qquad {}^{c}y_d = \frac{y_0 - {}^{i}y_d}{k_y} \tag{3.3} $$

obtaining a cubic equation, where k_x, k_y are the scaling factors in the *x* and *y* directions, respectively. They account for differences on the image axes scaling. The principal point of the image is defined by (x_0, y_0), and it represents the coordinates of the projection of the optical center of the camera over the image plane. ${}^{i}x_d$ and ${}^{i}y_d$ are the coordinates of the distorted point expressed in the image coordinate system.

Another widely used technique to solve equation (3.2) consists on applying the method of Tsai, setting ${}^{c}x_d$ and ${}^{c}y_d$ to:

$$ {}^{c}x_d = \frac{d_x\left(x_0 - {}^{i}x_d\right)}{s_x} \qquad {}^{c}y_d = d_y\left(y_0 - {}^{i}y_d\right) \tag{3.4} $$

where d_x, d_y are constant values computed from the parameters provided by the camera manufacturer, and s_x is the scaling factors in the *x* direction. This is the option taken by the researchers of the Instituto Superior Tecnico in [Gra97].

Negahdaripour *et al.* also take into account tangential distortion in order to compensate the distortions of the lenses [Xu00]. It can be computed by an infinite series [Wen92] that is normally approximated by one or two terms. However, tangential distortion is the responsible of a small percentage of the total lens distortion, and some authors consider that only the radial component should be computed to avoid numerical instability in calibration [Tsa87].

The calibration phase has to be performed underwater, since the medium properties can modify the camera parameters that would otherwise be measured out of the water. Figure 3.3 illustrates this effect.

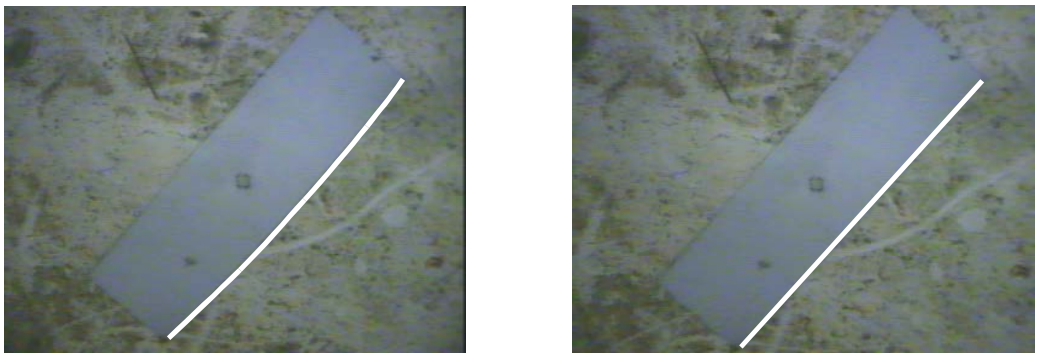


Figure 3.3. Correction of lens distortion in underwater images (a) Image with severe lens distortion; (b) corrected image. A white line has been overlayed on the contour of the object.

By correcting the lens distortion, the further steps of the mosaic construction will be more accurate and reliable –although some authors consider that the effect of lens distortion can be ignored. In this sense, the system presented in [Mar95] utilizes a camera with a narrow field-of-view (less than 20°). By using a camera with such a geometry, the perspective effects are minimized in two ways: first, data failing to accomplish the unique-plane condition assumed by the mosaic are less annoying; secondly, the effect of lens distortion can be ignored. Moreover, if we take into account the 3D nature of the underwater terrain, it can be seen that the overlapping area in two consecutive images is more similar as the field of view of the camera is reduced, since the image approaches an orthographic projection, in spite of being a perspective projection. However, as the field of view is small, more images are needed to cover the same area.

3.2.2 Lighting inhomogeneities and artifacts removal

Often, natural light is not sufficient for imaging the sea floor. For this reason, a light source attached to the submersible provides the necessary lighting. As well as the artifacts described in the previous section (scattering, absorption, etc.), the artificial light sources tend to illuminate the scene in a non-uniform fashion, producing a bright spot in the center of the image with a poorly illuminated area surrounding it. Also, brightness of the scene changes as the vehicle moves. Figure 3.4(a) shows a typical underwater frame suffering from this effect.

(a) (b)

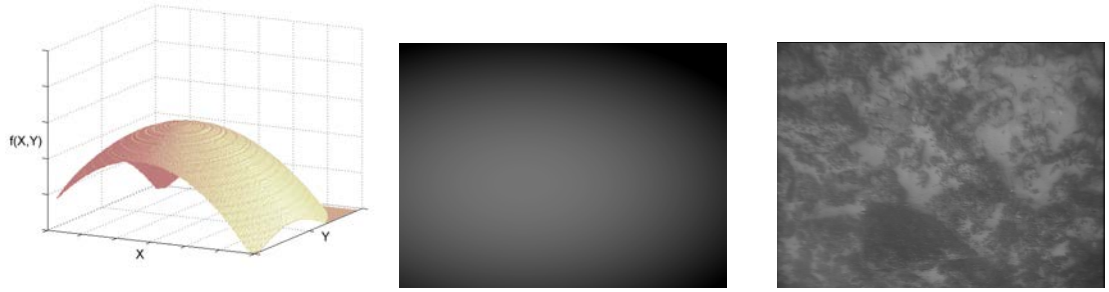
Figure 3.4. Underwater image showing lighting (a) inhomogeneities and (b) artifacts

Some authors have proposed the use of local equalization to compensate for the effects of non-uniform lighting [Sin98], darkening the center of the image and lighting the dark zones of the sides. Moreover, the motion of the light source creates a shift of the shadows induced in the scene. The motivation of the work described in [Sin98] was to eliminate differences in intensity among the component images that form the mosaic, thus enhancing the sense of continuity across the mosaic. This idea was later applied in [Eus00] to make the image irradiance more uniform before computing the registration parameters.

Rzhanov *et al.* presented in [Rzh00] a similar method for removal of lighting inhomogeneities: the so-called *de-trending* technique. It consists in the fit of a surface to every frame, and then subtracts it from the image. Knowledge about the nature of the light may suggest the best shape for the surface function. A two-dimensional polynomial spline is normally enough. Figure 3.5(a) illustrates the effect of correcting the image shown in Figure 3.4(a) by using this method.

Marks *et al.* [Mar95] adopted an alternative technique to deal with lighting non-uniformities. They proposed the use of a spatial filter that attenuates the lighting inhomogeneities: the Laplacian-of-Gaussian (LoG), proposed by Marr and Hildreth in [Mar80]. It was initially introduced as an edge detector, since it detects abrupt intensity variations in the image. It consists of a Gaussian smoothing of the image, which reduces the effect of noise on the image. When applied to underwater imaging, this low-pass filtering reduces the high-frequency artifacts on the image originated by backscattering, or “marine snow”, although it may also destroy part of the information in the rest of the image. For this reason the standard deviation of the

filter (σ) must be set up accurately. Next, a Laplacian operator performs a spatial second derivative on the image. According to Fleischer *et al.* this has the effect of separating the image into regions of similar texture. To obtain this effect, the filter requires the use of larger masks.



(a) (b) (c)

Figure 3.5. (a) spline surface fitted to the image of Figure 3.4a, (b) corresponding image, (c) correction of non-uniform illumination through spline subtraction as described in [Rzh00].

Balasuriya and Ura [Bal01] have also used a LoG filter to reduce backscatter, setting the mask size to 16×16. When both the Gaussian and the Laplacian filters are applied together, the result is a band-pass filter, with a band frequency that can be adjusted by means of the parameter σ of the Gaussian filter. Figure 3.6 shows the shape of the LoG filter with σ=4 and a size of 25×25 pixels.

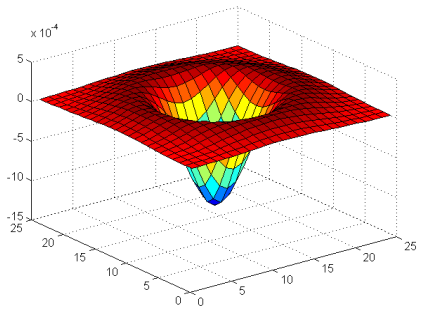


Figure 3.6. Laplacian-of-Gaussian convolution mask. Size: 25×25 pixels. $\sigma = 4$

In [Mar94b], the authors set the size of the filter to up to 40×40 pixels as typical values. Figure 3.7 illustrates the effect of convolving the images (6a) and (7c) with

the LoG filter. The binary image resulting from the SLoG filter is used in [Mar94b,Fle97] to register the image with the rest of the mosaic.

Negahdaripour *et al.* take a different approach. Applying the so-called Generalized Dynamic Image Model [Neg93] [Neg98c] solves the problem of light variation in the temporal domain. In this case, temporal radiometric differences on the image pixel values are taken into account by introducing two additional parameters into the constant-brightness optical flow equations. In order to provide a better global understanding, this method is explained in the next section, after the description of the Direct Estimation method.

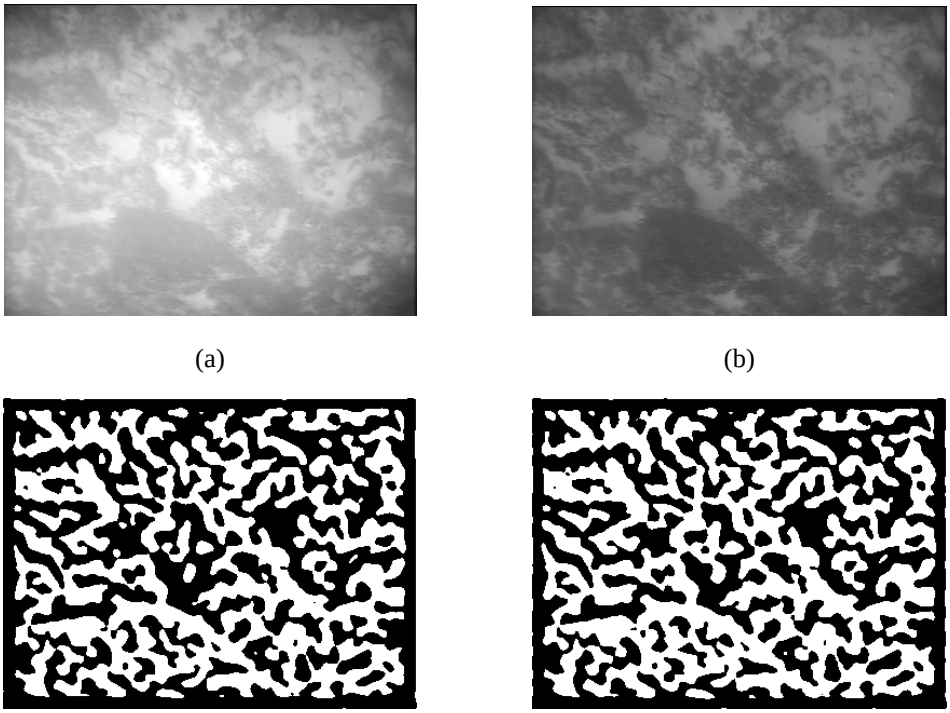


Figure 3.7. (a) and (b) gray-level images presenting different levels of illumination, (c) and (d) signum of Laplacian-of-Gaussian to the top images

3.2.3 Motion detection between consecutive frames

3.2.3.1 Introduction

This phase consists of detecting the apparent motion of the camera. This measurement will be computed in image coordinates. Most of the works that can be found in this study consider that the vehicle has 4-degrees of freedom: they assume that the vehicle is passively stable in roll and pitch, therefore only 3D translation and yaw motion is taken into account. This is a very reasonable assumption when the center of mass of the submersible is below its center of buoyancy. The literature exhibits two different approaches in the selection of the coordinate system to describe the vehicle motion. The first, assumes a coordinate system of the vehicle as commonly taken in underwater robotics [Sna50] (Figure 3.8a), while the second modifies the coordinate system to make it agree with the image plane of the camera (Figure 3.8b). In this case, the coordinate origin O_C is attached to the focal point of the camera, as normally considered in visual servoing tasks [Lot01].

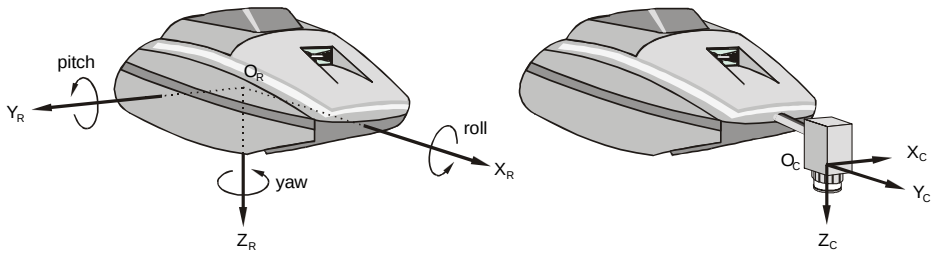


Figure 3.8. Cartesian coordinate system attached to the underwater vehicle and associated angles.

The motion detection techniques can be classified according to different parameters, as illustrated in Figure 3.9. The most general classification would distinguish between techniques working in the spatial or frequencial domain. Although it is known that a two-dimensional translation between two images can be determined by a shift of their Fourier spectrums, very few works have used this technique in order to construct a mosaic (only [Rzh00] to the best of our knowledge). The rotation and scaling parameters can be obtained by using the Fourier transform and a log-polar representation, then applying the Mellin transform. Moreover, some

authors have compared spatial and frequencial techniques applied to underwater image processing (*i.e.*, Olmos *et al.* in [Olm00]), concluding that spatial methods (feature-detection in their case) provided better results than the frequency-based methods on both synthetic and real images, "although not for a significant advantage". However, because frequencial techniques in underwater mosaicking systems have been used less in underwater mosaicking systems, they will not be described in detail in this report.

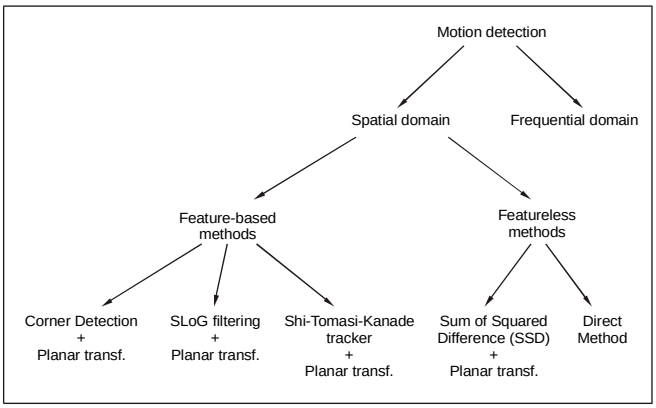


Figure 3.9. Classification of the main motion-detection techniques used in underwater mosaicking.

Spatial techniques can be further classified in two categories: *feature-based* and *featureless* methods. The first presume that feature correspondences between image pairs can be obtained, and utilize these matchings to find a transform which registers the image pairs. The last, on the contrary, minimize an energy function searching for the best transform without using any correspondences. In both cases, the aim can be reduced to the estimation of the parameters $h_{11}, h_{12}, ..., h_{33}$ of equation (3.5), where $\tilde{\mathbf{p}}^{(k)} = (x_i^{(k)}, y_i^{(k)}, 1)^T$ and $\tilde{\mathbf{p}}^{(k+1)} = (x_i^{(k+1)}, y_i^{(k+1)}, 1)^T$ denote a correspondence point in the images taken at time *k* and *k*+1, respectively, expressed in homogeneous coordinates. The equations for perspective projection to the image plane are non-linear when expressed in non-homogeneous coordinates, but are linear in homogeneous coordinates. This is characteristic of all transformations in projective geometry, not just perspective projection. It provides one of the main motivations for the use of homogeneous coordinates, since linear systems are symbolically and numerically easier to handle than non-linear ones.

$$\tilde{\mathbf{p}}^{(k)} = {}^{k}\mathbf{H}_{k+1} \cdot \tilde{\mathbf{p}}^{(k+1)} \quad \text{or} \quad \begin{bmatrix} x_i^{(k)} \\ y_i^{(k)} \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i^{(k+1)} \\ y_i^{(k+1)} \\ 1 \end{bmatrix} \tag{3.5}$$

The estimation of $h_{11}, h_{12}, ..., h_{33}$ is known as image registration, and these 9 parameters relate the coordinates of two images in the sequence (determining a projective transform). The symbol $\cong$ indicates equality up to scale.

Matrix ${}^{k}\mathbf{H}_{k+1}$ represents a projective transform describing the inter-frame motion. Some refer to this matrix as a *homography* (or collineation) [Sze94]. The homography matrix ${}^{k}\mathbf{H}_{k+1}$ relates the 2D coordinates of any point of image $I^{(k+1)}$ with the coordinates of the same point expressed in the reference frame of image $I^{(k)}$.

3.2.3.2 Feature-based techniques

In terrestrial environments, typical features to track are points, lines or contours. As pointed out in section 2, straight lines and contours are normally difficult to find in the underwater environment. The feature-based methods normally solve the registration by first detecting image corners or highly textured patches in one image $(x_i^{(k)}, y_i^{(k)})$, and then matching them through correlation on the next image $(x_i^{(k+1)}, y_i^{(k+1)})$, or minimizing a cost function, considering in both cases that the same scene radiance is kept constant through the image sequence. Generally, images are low-pass filtered before correlation, since correlation strength is sensitive to noise [Gia00].

One of the first feature-based mosaicking systems was developed by MBARI/Stanford researchers [Mar95, Fle96, Fle00]. They locate features with a large image gradient (*i.e.* contours) through the use of the "Laplacian of the Gaussian" (LoG) operator. Instead of correlating brightness values, a binary image is obtained depending on the resulting signum of LoG. It simplifies the correlation to a XOR operation. Moreover, as described in section 0, this method provides some degree of robustness with respect to artifacts, due to non-uniform illumination.

The researchers of Heriot-Watt/Udine Universities [Tom98, Fus99, Odo99, Pla00] select the features to compute image registration by means of the Shi-Tomasi-Kanade tracker [Shi94]. In this way, given a point $\mathbf{p}_i$ for which the motion is to be estimated, a small region R_i centered at this point is considered. Then, the matrix of the partial derivatives **G** is computed as follows:

$$\mathbf{G} = \sum_{R_i} \begin{bmatrix} I_u^2 & I_u I_v \\ I_u I_v & I_v^2 \end{bmatrix}, \text{ with } I_u = \left(\frac{\partial I}{\partial x} \right) \text{ and } I_v = \left(\frac{\partial I}{\partial y} \right) \tag{3.6}$$

A feature point $\mathbf{p}_i$ is a good candidate to track if **G** is well conditioned, that is, if both eigenvalues of **G** are above a user-defined threshold. This means that the image point $\mathbf{p}_i$ presents a rapid intensity variation on neighboring pixels in the *x* and *y* directions. The entire image is scanned, searching for good candidate points $\mathbf{p}_i$. This approach can be compared to the detection of corner points [Har88], since it enhances regions with a high spatial frequency content in both *x* and *y* directions.

Considering that the time sample frequency is sufficiently high, the intensities of every interest point and its neighboring pixels can be considered to remain unchanged in two consecutive images, as introduced by the *Brightness Constancy Model* in [Hor86]:

$$I^{(k)}(x, y) = I^{(k+1)}(x + \Delta x, y + \Delta y) \tag{3.7}$$

In this way, the motion is approximated by a simple translation $\mathbf{d} = (\Delta x, \Delta y)$. Since the assumed motion model is not perfect, and the image irradiance may not remain constant, the problem is rearranged as finding the displacement **d**, which minimizes the SSD residual:

$$\varepsilon = \sum_{R_i} \left[I^{(k+1)}(x + \Delta x, y + \Delta y) - I^{(k)}(x, y) \right]^2 \tag{3.8}$$

If the image motion is assumed to be small, the term $I^{(k+1)}(x + \Delta x, y + \Delta y)$ can be approximated by its Taylor series expansion, truncated to the linear term, and imposing that the derivatives with respect to **d** are zero:

$$\varepsilon \approx \sum_{R_i} \left[I^{(k)}(x_0, y_0) + \left[I_u, I_v \right] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} + I_t \tau - I^{(k)}(x_0, y_0) \right]^2 \tag{3.9}$$

where $I_t = \left(\partial I / \partial t \right)$ and τ is the elapsed time between images $I^{(k)}$ and $I^{(k+1)}$. Operating the terms in equation (3.9):

$$\varepsilon(\Delta x, \Delta y) \approx \sum_{R_i} \left[\left(\left[I_u, I_v \right] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \right)^2 + 2 \left[I_u, I_v \right] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} I_t \tau + \left(I_t \tau \right)^2 \right] \tag{3.10}$$

Then, deriving ε with respect to $\mathbf{d} = (\Delta x, \Delta y)$, imposing $\partial \varepsilon / \partial \Delta x = 0$ and $\partial \varepsilon / \partial \Delta y = 0$, and operating we obtain:

$$\sum_{R_i} \left[2 \begin{bmatrix} I_u \\ I_v \end{bmatrix} \left[I_u, I_v \right] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} + 2 \left[I_u, I_v \right] I_t \tau \right] = 0 \tag{3.11}$$

which can be arranged as:

$$\left(\sum_{R_i} \begin{bmatrix} I_u^2 & I_u I_v \\ I_u I_v & I_v^2 \end{bmatrix} \right) \mathbf{d} + \left(\sum_{W_i} I_t \begin{bmatrix} I_u \\ I_v \end{bmatrix} \right) \tau = 0 \tag{3.12}$$

Then, the following linear system can be obtained [Odo99]:

$$\mathbf{G} \cdot \mathbf{d} = -\tau \sum_{W_i} I_t \begin{bmatrix} I_u \\ I_v \end{bmatrix} \tag{3.13}$$

The displacement vector $\mathbf{d}$ can be computed for every selected point $\mathbf{p}_i$ through an iterative Newton-Raphson scheme that minimizes equation (3.13).

Another feature-based method can be found in [Guo00]. Although the authors do not describe how they select the features, the most likely trajectory is computed for every feature by means of a least-mean squared-error estimator and a Kalman Filter, in the framework of a Maximum a Posteriori estimation technique. This approach can be compared with the previous one in the sense that it also tracks a certain number of features within a sequence, with the aim of constructing a visual map. In [Guo00] the features are assumed to undergo 2D translations and rotations in a plane parallel to the image.

We have described until now how to match feature points in two consecutive images. However, this idea was later extended to compute the motion between the first frame of the sequence (known as “reference frame”) and every incoming image, tracking point features over longer sequences [Tom98, Tru00a, Tru00b, Odo99]. For this reason, the translational consecutive-frame displacements, proposed initially in [Tom91], have been extended to an affine model, which could cope with more

complex motions over longer sequences [Shi94]. In this way the feature window can undergo rotation, scaling and shear in addition to translation, and the affine model can be used to monitor the quality of the tracking. Nevertheless, when constructing a mosaic, the initially tracked features may disappear from the field of view. In this case, a new reference image has to be selected and new features are chosen for tracking.

Once the Shi-Tomasi-Kanade tracker has detected a set of correspondences $\left(\mathbf{p}_i^{(k)}, \mathbf{p}_i^{(k+n)}\right)$ relating two images (consecutive or not), the nine unknown parameters $h_{11}, h_{12}, ..., h_{33}$ can be found by solving the following rank-deficient system of homogeneous linear equations [Odo99]:

$$\mathbf{U} \cdot \mathbf{h} = 0 \tag{3.14}$$

$$\begin{bmatrix} x_1^{(k)} & y_1^{(k)} & 1 & 0 & 0 & 0 & -x_1^{(k)} \cdot x_1^{(k+n)} & -y_1^{(k)} \cdot x_1^{(k+n)} & -x_1^{(k+n)} \\ 0 & 0 & 0 & x_1^{(k)} & y_1^{(k)} & 1 & -x_1^{(k)} \cdot y_1^{(k+n)} & -y_1^{(k)} \cdot y_1^{(k+n)} & -y_1^{(k+n)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^{(k)} & y_n^{(k)} & 1 & 0 & 0 & 0 & -x_n^{(k)} \cdot x_n^{(k+n)} & -y_n^{(k)} \cdot x_n^{(k+n)} & -x_n^{(k+n)} \\ 0 & 0 & 1 & x_n^{(k)} & y_n^{(k)} & 1 & -x_n^{(k)} \cdot y_n^{(k+n)} & -y_n^{(k)} \cdot y_n^{(k+n)} & -y_n^{(k+n)} \end{bmatrix} \cdot \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0 \tag{3.15}$$

Equation (3.14) can be obtained by expanding equation (3.5) for the case where several point matches are available between images $I^{(k)}$ and $I^{(k+n)}$. Equation (3.15) is solved in [Odo99] through Singular Value Decomposition (SVD), after imposing the constraint of unit norm for **h**.

The computation of equation (3.15) requires at least four pairs of corresponding points $\left(\mathbf{p}_i^{(k)}, \mathbf{p}_i^{(k+n)}\right)$, as long as collinearity between any 3 points is avoided. Normally, more than 4 points are used, obtaining an over-determined system of equations solved through a least squares approach.

Another feature-based strategy is used by Gracias and Santos-Victor [Gra97,Gra98b,Gra99,Gra00], who detect features in image $I^{(k)}$ by means of a slightly modified version of the Harris corner detector [Har87]. Then, the corresponding matches in the next image $I^{(k+1)}$ are obtained through correlation. In the correlation phase, they obtain sub-pixel accuracy by means of an optical-flow technique applied to the patches around each corner [Gra98a]. It is also possible to obtain sub-pixel accuracy by estimating the peak location of the cross-correlation, and then fitting a parametric surface to the location of every corner [Mar95]. Once the correspondences are available, matrix ${}^{k}\mathbf{H}_{k+1}$ is computed following the same strategy as described above.

When a new image has to be added to the mosaic, ${}^{k}\mathbf{H}_{k+1}$ provides its best fitting with respect to the previous image (or to the mosaic image). The most general homography has 8 free parameters and is known as projective transformation (translation, rotation, scaling, and perspective deformation). Since projective transformation can be expressed in terms of 8 degrees of freedom, it may not be the best way to describe a given motion, and a better motion model can be assumed.

As described in [Gra00], if the sort of camera motion is known beforehand, the projective model may contain more free parameters than necessary. The simplest transformation is pure translation, followed by translation and rotation (rigid or Euclidean model) and next a more complicated motion model can be described by introducing scaling (similarity model). More complex transformations are obtained with the affine model (translation, rotation, scaling, and shear). Finally, the projective transformation introduces the perspective deformation to the affine transformation. Table 1 shows the homography representing some of the most popular transformations. Depending on the nature of the motion, the most suitable motion model will provide the best results in the image registration phase. The problem is that in general, it is difficult to know the motion model that best describes the motion of the vehicle beforehand.

rigid transformation	**affine transformation**	**projective transformation**
translation and rotation	translation, rotation, scaling and shear	translation, rotation, scaling and perspective deformation
$\begin{bmatrix} x_i^{(k)} \\ y_i^{(k)} \\ 1 \end{bmatrix} \cong \begin{bmatrix} \cos\theta & -\text{sen}\theta & t_x \\ \text{sen}\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i^{(k+1)} \\ y_i^{(k+1)} \\ 1 \end{bmatrix}$	$\begin{bmatrix} x_i^{(k)} \\ y_i^{(k)} \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{11} & h_{12} & t_x \\ h_{21} & h_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i^{(k+1)} \\ y_i^{(k+1)} \\ 1 \end{bmatrix}$	$\begin{bmatrix} x_i^{(k)} \\ y_i^{(k)} \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x_i^{(k+1)} \\ y_i^{(k+1)} \\ 1 \end{bmatrix}$

Table 3.1: Possible motion models for planar transformations.

As can be observed in Table 3.1, the most general planar transformation has eight independent parameters (projective model). In the case of underwater imaging, additional constraints can be available on the camera motion. For instance, if the vehicle is known to be passively stable in pitch and roll, the parameters h_{31} and h_{32} of the projective model can be set to zero, since no perspective deformation will occur on the image, obtaining the affine transformation of table 1. Therefore, the simplest 3x3 matrix that fits the motion of the camera will be the best approximation of its described trajectory, with respect to more complex motion models.

Improving image registration

Before the computation of the homography that registers two consecutive images, better results can be obtained by analyzing the data that is used to find matrix ${}^{k}\mathbf{H}_{k+1}$. Some of the homography-based mosaicking systems (*i.e.* [Gra99, Gra00, Odo99, Gar01]) reduce the amount of "outliers" (data describing a movement in gross disagreement with the general motion) by applying robust techniques to the pairs point-matching [Rou87,Mee91].

A widely-used technique for detecting outliers is the *Least Median of Squares* (LMedS) algorithm [Rou87]: given the problem of computing the homography matrix ${}^{k}\mathbf{H}_{k+1}$ from a set of data points, where *n* is the minimum number of data points which determine a solution, compute a candidate solution based on a randomly chosen *n*-tuple from the data. Then, estimate the fit of this solution to all the data,

defined as the median of the squared residuals. The median of the squared residuals is defined by:

$$M_{err} = \underset{j}{med}\left(d^2\left(\tilde{\mathbf{p}}_j^{(k)}, {}^k\mathbf{H}_{k+1}\tilde{\mathbf{p}}_j^{(k+1)}\right)\right) + \left(d^2\left(\tilde{\mathbf{p}}_j^{(k+1)}, {}^k\mathbf{H}_{k+1}^{-1}\tilde{\mathbf{p}}_j^{(k)}\right)\right) \qquad (3.16)$$

where $\tilde{\mathbf{p}} = (x_1, x_2, x_3)$ are the homogeneous coordinates of a 2D point $\mathbf{p}$ defined in the image plane; and $d^2(\tilde{\mathbf{p}}_j^{(k)}, \mathbf{H}\tilde{\mathbf{p}}_j^{(k+1)})$ is the square distance from a point $\tilde{\mathbf{p}}_j^{(k)}$, defined on image $I^{(k)}$, to the projection on the same image plane of its correspondence $\tilde{\mathbf{p}}_j^{(k+1)}$. Once the best solution has been found, a minimal median is obtained. As from the median, the mean μ and the standard deviation σ can be computed (see [Rou87] for details). Therefore, those points at a distance *d* larger than $\mu \pm \sigma$ are eliminated, and matrix ${}^k\mathbf{H}_{k+1}$ is recomputed with the remaining points, through a Least Squares criteria. This outlier rejection process is called Dominant Motion Estimation in [Odo99].

According to [Odo99], when Gaussian noise is present the relative statistical efficiency of LMedS can be increased by running a weighted Least Squares fit after LMedS. In this case, weights are selected depending on the residual of the LMedS procedure [Rou87].

Gracias and Santos-Victor propose a two-step variant of LMedS, known as *MEDian SEt REduction* (MEDSERE) [Gra98a,Gra00]. It consists on two iterations of LMedS random sampling, choosing the best data points in fitting the cost function of equation (3.16). This technique requires less random sampling than LMedS while obtaining the same degree of outlier rejection.

Tommasini *et al.* [Tom98] devised a method called X84 to automatically reject incorrect matching points in the image sequence, as initially proposed in [Ham86]. Their method is based on a measurement of the residual of the match between the initial image and every frame of the sequence. A tracked feature is considered to be good (reliable) or bad (unreliable) according to this residual.

Moreover, in addition to the techniques described above, a better-conditioned problem can be obtained if the data undergoes a standarization process [Gra98a], achieving more accurate results. A typical standarization consists on placing the

coordinate center of the images in the centroid of the data points. Then, the points are re-scaled, so that the average distance from the center to all the points is $\sqrt{2}$.

3.2.3.4 Feature-less techniques

An alternative to the methods described above is the computation of motion without the need to estimate feature correspondences. Two approaches fall into the feature-less group: (1) Homography-based global minimization techniques, which compute a 2D (projective) transformation matrix; and (2) a direct method –derived from the optical flow equation– that is able to provide 3D motion estimation without the need of intermediate 2D computations.

Estimating the 2D transformation through global minimization

The feature-less techniques minimize the sum of the squared intensity errors over all corresponding pairs of pixels, which are present in two consecutive images, as was shown in equation (3.8), but this time the window W_i is extended to the whole image:

$$\varepsilon = \sum_i \left[I^{(k+1)}(x_i^{(k+1)}, y_i^{(k+1)}) - I^{(k)}(x_i^{(k)}, y_i^{(k)}) \right]^2 \qquad (3.17)$$

where $I^{(k)}$ and $I^{(k+1)}$ represent the images taken at time instant k and k+1, respectively.

The equation, which relates the pixels in both images by means of an homography (equation (3.5)), is then taken as a cost function that minimizes the matching criterion ε of equation (3.17), in order to obtain the parameters $h_{11}, h_{12}, ..., h_{33}$ through a nonlinear minimization technique.

Acquisition and matching of good features for motion detection is a difficult task in underwater images. In this respect, feature-based methods are error-prone. However, since featureless methods do not rely on explicit feature correspondences, they suffer no problems associated with feature detection and tracking. Nevertheless, these methods require good initialization values in order to converge to a solution. Moreover, they need a small change from one image to the next. But, even when the motion between images is smooth, there is no guarantee that the parameter estimate

process will lead to the optimal solution. Special efforts must be made to prevent the parameter estimation from falling into local minima. Finally, when dealing with the computation of the homography matrix, the computational requirements of the featureless methods is higher than that of the feature-based approach.

For all the reasons explained above, homography-based feature-based methods are more popular than featureless ones in sub-sea mosaicking applications.

Direct method of computing 3D motion

Direct motion estimation methods are based on the following statement: if the aim of the mosaicking system is to obtain the 3D motion of the vehicle, it is not necessary to first compute the 2D image motion and then to use this estimation to obtain the 3D measure. The use of spatio-temporal image gradients $(I_u, I_v, I_t) = \left(\partial I/\partial x, \partial I/\partial y, \partial I/\partial t\right)$ allows the computation of the 3D motion directly. Negahdaripour *et al.* derive their solution of the motion problem by applying the *Brightness Constancy Model* (BCM) [Hor86]. Then, revisiting equation (3.7), the BMC assumes that a pixel located at coordinates (x,y) in one image conserves its brightness when located at position $(x + \Delta x, y + \Delta y)$ in the next image.

Considering $\Delta x = u\delta t$ and $\Delta y = v\delta t$, equation (3.7) can be re-written as:

$$I^{(k+1)}(x + u\delta t, y + v\delta t) = I^{(k)}(x, y) \tag{3.18}$$

where (*u*,*v*) are the image velocity components of the pixel at time (*k*) in the *x* and *y* directions, respectively.

Again, applying Taylor series expansion to equation (3.18), the *optical flow* equation is obtained:

$$I_u u + I_v v + I_t \approx 0 \tag{3.19}$$

If the image motion is expressed in terms of the translational (t_x, t_y, t_z) and yaw (Ψ) motion of the vehicle, equation (3.20) can be derived.

$$I_t + \left[\begin{pmatrix} -f & 0 \\ 0 & -f \\ x_i & y_i \\ y_i & -x_i \end{pmatrix} \cdot \begin{pmatrix} I_u \\ I_v \end{pmatrix} \right]^T \cdot \begin{bmatrix} t_x/Z \\ t_y/Z \\ t_z/Z \\ \Psi \end{bmatrix} \approx 0, \text{ for } i = 1..n \tag{3.20}$$

where f is the focal length of the camera (obtained through calibration), Z is the average vertical distance to the sea floor at time instant k; (t_x, t_y, t_z, Ψ) represent the 3D translation and yaw rotation of the vehicle (the only unknowns in the equation); and (x_i, y_i) are the image coordinates of any pixel i. Equation (3.20) holds for all the image, with (x_i, y_i) and I_t varying for every pixel. This equation is applied to n pixels, solving the following system for the 4 unknowns through a least squares method:

$$\begin{bmatrix} t_x/Z \\ t_y/Z \\ t_z/Z \\ \Psi \end{bmatrix} = -\left[\sum_i \left(ss^T \right) \right]^{-1} \left[\sum_i \left(sI_t \right) \right], \text{ where } s = \begin{pmatrix} -f & 0 \\ 0 & -f \\ x_i & y_i \\ y_i & -x_i \end{pmatrix} \cdot \begin{pmatrix} I_u \\ I_v \end{pmatrix} \tag{3.21}$$

The solution of this system is constrained to a relatively flat bottom, as has also been assumed by the homography-based methods. Therefore, the local differences in depth all over the image should be insignificant relative to the average distance Z from the vehicle to the seabed. The initial altitude $Z^{(0)}$ may be acquired from a sonar reading, and can be considered to update Z at every time step.

Although equations (3.18) to (3.21) have been described for the constant illumination case, Negahdaripour *et al.* [Neg99] have proved that temporal radiometric differences on the image pixel values can be taken into account by introducing two additional parameters: a multiplying factor m and an offset c. This approach is based on the so-called *Generalized Dynamic Image Model* (GDIM) [Neg93,Neg98c]. The radiometric transformation fields m and c are considered low frequency spatial signals that explain the instantaneous rate of image irradiance variation between a point (x,y) in one image with the same point $(x + \Delta x, y + \Delta y)$ in the next image of the sequence, as shown in equation (3.22).

$$I^{(k+1)}(x + u\delta t, y + v\delta t) = m \cdot I^{(k)}(x, y) + c \tag{3.22}$$

This equation can also be expanded to a Taylor series up to the first order terms, expressing the two unknowns (*u*,*v*) in terms of the vehicle motion $\left(t_x, t_y, t_z, \Psi\right)$:

$$I_t + \left[\begin{pmatrix} -f & 0 \\ 0 & -f \\ x_i & y_i \\ y_i & -x_i \end{pmatrix} \cdot \begin{pmatrix} I_u \\ I_v \end{pmatrix}\right]^T \cdot \begin{bmatrix} t_x/Z \\ t_y/Z \\ t_z/Z \\ \Psi \end{bmatrix} + \left[I^{(k)}(x_i, y_i) \quad 1\right] \cdot \begin{bmatrix} 1-m \\ c \end{bmatrix} \approx 0, \text{ for } i = 1..n \qquad (3.23)$$

Although equation (3.23) holds for all the image points, parameters *m* and *c* vary (smoothly) through the image. Bearing in mind this low frequency characteristic, Negahdaripour *et al.* [Neg99] divide the image in small regions R_i, assuming a constant value of *m* and *c* within these regions.

Direct methods [Neg99] present some advantages over optical flow or feature correspondences, such as a lower computational cost, higher accuracy, and the possibility of taking into account radiometric variations. For this reason it can be efficiently implemented for achieving real-time performance.

3.2.4 Mosaic registration and actualization

Section 3.2.4 has described the methods to detect motion between consecutive images. To facilitate the detection of correspondences between images, some of these methods constrain the inter-frame motion to a small value. Obviously, small errors in the detection of motion between consecutive frames provoke an accumulated error as the mosaic increases in size. This error can be reduced if the current frame is periodically registered with the mosaic image, as will be described in this section.

Once a first estimate of the image registration parameters is known, the mosaicking system has to decide when, either it is worth actualizing the mosaic with the present image, or it does not pay to update the mosaic because the contribution of the present image to the mosaic is too small. Three criteria can be used to take the decision of updating the mosaic: (i) use all the registered images to update the mosaic; (ii) update at constant time intervals; and (iii) update at constant displacement intervals.

Strategy (i) is used by the Oceans Systems Lab [Odo99,Tru00b] and IST [Gra98a,Gra00] researchers, capturing the images at quite a high frequency (close to video rate), and then processing them offline. In general, this presents the advantage of providing a rich amount of information, allowing the use of temporal filtering to segment moving objects from the stationary background, as will be described later in section 3.2.5. We could consider that Rhzanov *et al.* [Rhz00] also uses this technique, although their approach is slightly different due to a lower capture rate (2-3 fps), thus presenting a smaller overlap between consecutive images.

Strategy (ii) is taken by Negahdaripour *et al.* [Neg98a,Neg98e]. In order to operate in real time, updating the mosaic with every new image of the sequence implies a loss of computational efficiency. For this reason, a constant parameter L governing the actualization rate of the mosaic is set in [Neg99]. In this way, the mosaic is updated with a new image every L frames. Parameter L is adjusted depending on the motion of the vehicle and its distance to the sea floor. When an image is selected to actualize the mosaic, an “a priori” estimation of the location of the image in the mosaic is computed through the registration of this image with the previous one. Then, an image is extracted from the mosaic in the estimated location and refined motion estimation is performed, thus reducing the accumulated error.

Strategy (iii) has been selected by MBARI/Stanford researchers as a good solution to obtain real-time performance [Mar95,Fle98]. Their system captures images at 30 Hz. Then, every image is registered with the last image added to the mosaic. The image may be selected to be part of the mosaic (“acquired”) depending on its overlapping region with some previously acquired image. Marks *et al.* consider that a new image is “acquired” only when it is fed into the composite mosaic image and not when it is snapped by the camera. The live image is acquired only if the horizontal and vertical image offsets are close enough to a desired set of offsets. In this way, the amount of images composing the mosaic is kept to a small value in relation to the mapped area. The authors consider that camera rotation and scaling can be considered to be small since the special-purpose hardware allows a high enough cycle time for processing the images. So this motion will not significantly degrade correlation and the subsequent registration.

Once the frame-to-frame motion parameters have been obtained, these transformations are combined to form a global model. The global model takes the form of a global registration, where all the frames are mapped into a common, arbitrarily chosen, reference frame.

The last step consists on merging together the registered (aligned) images, in order to create a mosaic. Some of the works in the literature refer to this step as mosaic reference. Once the best transformation ${}^{k}H_{k+1}$ has been found, images $I^{(k+1)}$ and $I^{(k)}$ can be *warped* together, but a base frame is necessary as an initial coordinate system. Some approaches use the first image of the sequence as an initial coordinate system, while other approaches map the first image into an arbitrarily chosen reference frame. This second approach was introduced in [Gra98a], where the mosaicking system was able to handle severe violations of the assumption of the camera being parallel to the ocean floor. In this way, every live image of the sequence can be registered with a virtual reference frame, as the one illustrated in Figure 3.10.

Once the first image is attached to the mosaic, the following images have to be registered not only to the previous image of the sequence, but also to the reference frame. This process of global registration relates the image coordinates of any point in image $I^{(k+1)}$ with respect of the coordinate frame of image $I^{(1)}$.

The global registration matrix ${}^{1}H_{k+1}$ is computed by multiplying the set of transformation matrices:

$$ {}^{1}H_{k+1} = \prod_{i=1..k} {}^{i}H_{i+1} \tag{3.24} $$

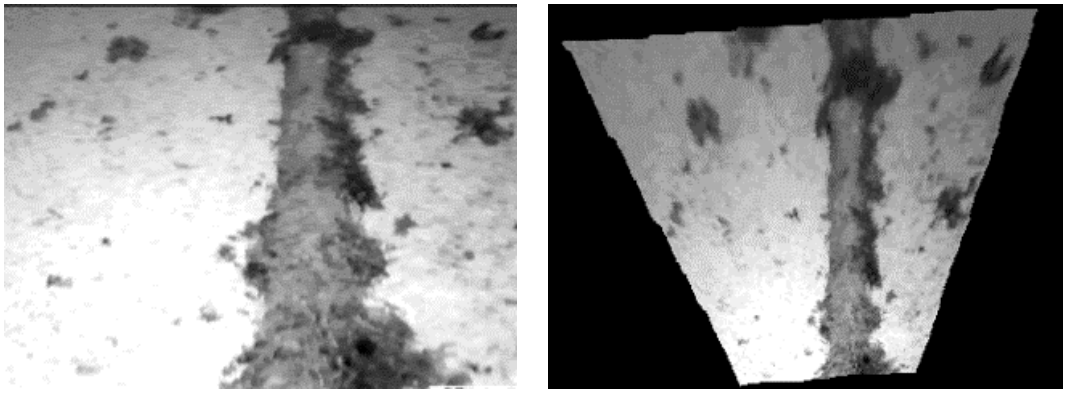


Figure 3.10. Arbitrary-chosen reference frame to obtain a better perception of the sea floor. It simulates the effect of having the camera parallel to the floor. Bilinear interpolation has been applied to the warped image.

3.2.5 Image warping and mosaic construction

Once the frame-to-mosaic motion parameters are known, the registered images are merged into a composite mosaic image. Then, a same region of the scene is viewed from different images, generating an overlapping area in the mosaic. As illustrated in Figure 3.11, the set of pixels in the registered images belonging to the same output point can be though of as lying on a line which is parallel to the time axis [Gra98a]. Several temporal filters can be used to "compose" the mosaic image on the overlapping regions. They can be divided into two main approaches: (a) Every mosaic pixel is obtained by combining the overlapping pixels; and (b) Only one of the aligned images is taken into account

Method (a) requires accurate alignment over the entire image, otherwise the resulting mosaic will present some blurred zones. Method (b) requires alignment only along the seams. Some implementations, in trying to obtain a uniform-looking mosaic image, also disguise the lighting differences along the seams through some sort of correction of the lighting inhomogeneities.

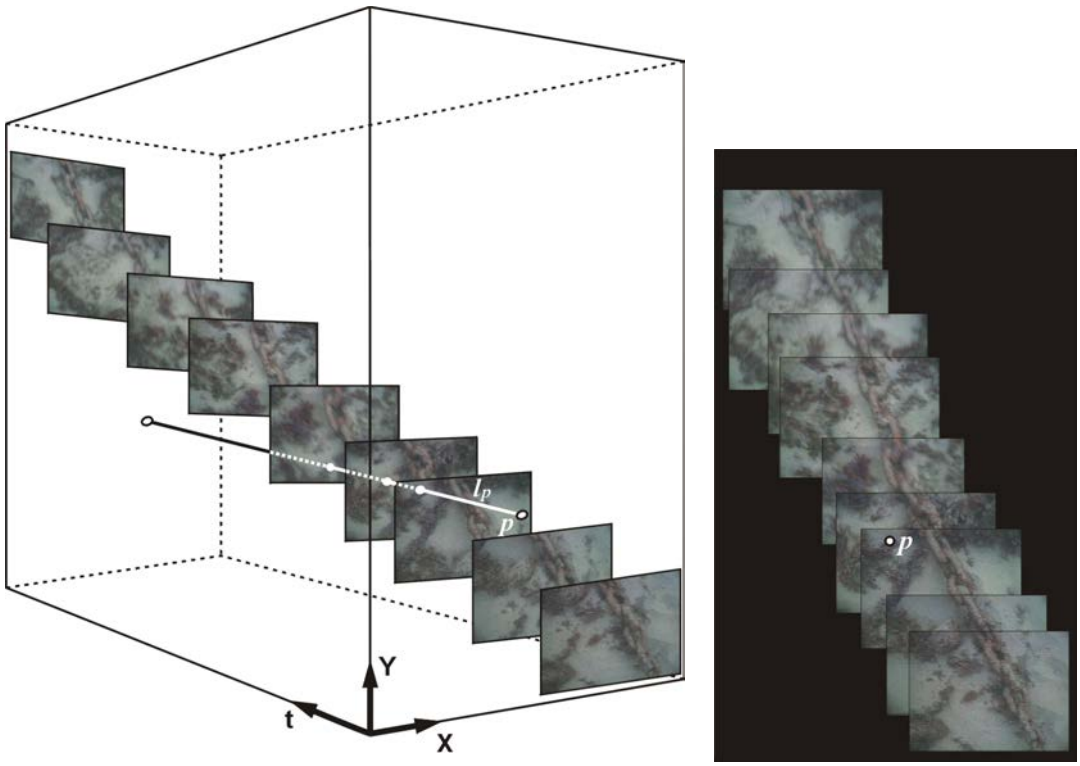


Figure 3.11. Space-time volume defined by the aligned images forming the mosaic. The line l_p, going along the temporal axe, intersects pixels that correspond to the same world point (in absence of parallax).

The combination of pixels that overlap in time can be performed by means of different strategies: (a1) temporal average, (a2) temporal median, (b1) most recent pixel or (b2) less recent pixel. Temporal average attenuates the presentation of fast moving objects (for example, fishes) onto the motionless background, generating a slight blurring on the areas where the object has moved. Temporal median solves more effectively this problem, but it is especially useful in the case of moving objects that occupy background pixel-coordinates during less than half the frames. These two temporal filters compromise the real-time performance of the mosaicking systems, all at the time of demanding more memory resources in order to construct the overlapping structure.

For all the reasons described above, the mosaicking systems that perform in real-time normally choose strategy (b): taking into account only one of the aligned pixels. In this case, one can select either the most recent information (called “use-last” in

[Gra98a]) to update the mosaic, or the less recent information (or “use-first”), which implies that every new image only actualizes the mosaic on that zones that have not been updated before. Negahdaripour *et al.* have selected this last strategy in order to obtain real-time performance [Neg99,Neg96]. In other respect, Odone and Fusiello proposed in [Odo99] two more temporal filters: (a3) weighed temporal median and (a4) weighed temporal average. In this case, weight decreases with the distance of the pixel from the image center. Figure 3.12 illustrates a classification of the temporal filters.

MBARI/Stanford researchers call to this phase the *consolidation* process. It uses the registration parameters to determine how to fuse the acquired image to the mosaic, but only the images that provide enough new information are consolidated into the mosaic.

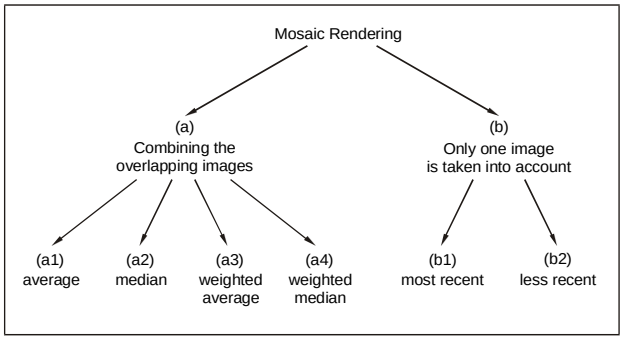


Figure 3.12. Classification of temporal filters to render the mosaic image.

3.2.6 Mosaic-driven navigation

In order to analyze the visual mosaicking systems, one further aspect has to be taken into account. According to the obtained information related to the mosaic, the control system has to decide how should the vehicle move to achieve the task of mosaic construction in the most adequate manner. This vision-based vehicle control allows the development of path-planning algorithms that aid the construction of mosaics. In this way, the vehicle can revisit a zone that has been already surveyed when the system detects that a gap has been left in the mosaic. Moreover, some authors have proved through simulation [How00] that the best results in the construction of visual

mosaics are obtained when the vehicle trajectory can be governed by the mosaicking system.

A commonly used technique to construct seabed visual mosaics consists of *consecutive-image mosaicking*. By using this strategy, every new image is registered with the last image that was added to the mosaic. This technique is also known as *single-column mosaic* and it is the strategy followed by most of the systems surveyed in this work (*i.e.* [Gra98a, Xu97, Rzh00, Mar94d]). It is obvious that every time an image is consolidated into the mosaic, there is a chance for error in the parameters registering this image to any other image. Therefore, considering a sequence of n images (from $I^{(0)}$ to $I^{(n-1)}$), the total error accumulates with every new image consolidated into the mosaic, obtaining an error o(n) if the n images are used to update the mosaic. However, significant differences arise among the works that use this technique, as was pointed out in section 0. While IST [Gra00] and Heriot-Watt [Tru00b] researchers use all the images of the sequence to generate the mosaic, MBARI/Stanford mosaicking system adds a new image $I^{(k)}$ to the mosaic only when the overlapping region with the previously added image is small enough. This operation is called "acquisition" in [Mar95]. This strategy reduces the drift error to o(m), where m is the amount of images that have been consolidated into the mosaic (with $m<n$). Negahdaripour *et al.* reduce the error to o(n/L) by registering the present image with the mosaic image every L images.

In order to map a wide area of the ocean floor, sonar-scan mapping systems have been using *column-relative* path planning for several years. This idea was applied to visual mapping by Marks *et al.* in [Mar94d, Mar95], where the new image is registered to the contiguous image of the previous column. In this way, the construction of a square mosaic formed by n images reduces the accumulated error to $\mathrm{O}\left(\sqrt{n}\right)$. The column-relative mosaic described in [Mar94d] could be accomplished in real time thanks to a specific hardware for image processing, although an additional constraint was introduced to simplify the registration phase: images had to be acquired at the same orientation. Thereby, the motion of the vehicle is restricted to a column, where the vehicle heading has to be kept constant, as shown in Figure 3.13. A significant contribution of [Mar95] was the demonstration of the possibility of creating mosaics of the ocean floor in real time, as the vehicle was moving. This

strategy allowed a proper image acquisition in order to avoid visual gaps in the mosaic. However, on-line processing allows the mosaic data to be used immediately for vehicle navigation.

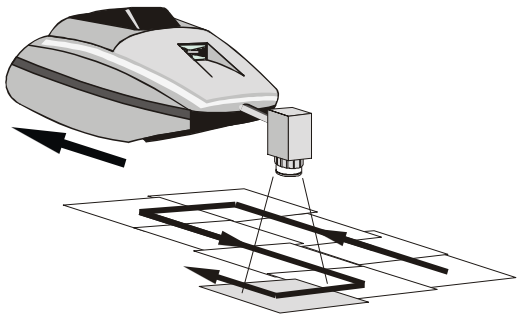


Figure 3.13. Multiple-column mosaicking system of [Mar95]. The system requires the vehicle to move forwards and backwards without altering its heading

The acquisition strategy of MBARI/Stanford researchers of [Mar95] was taken on step forward in [Fle97], defining which previously acquired image has to be correlated to every new live image (see Figure 3.14).

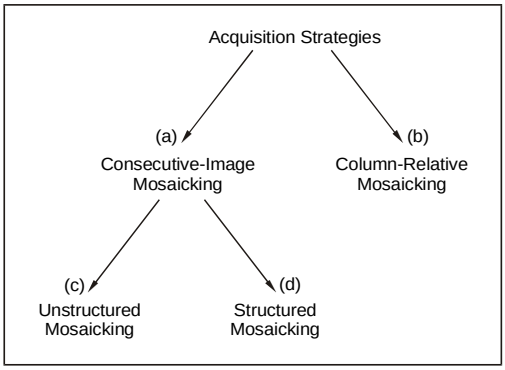


Figure 3.14. The acquisition module governs how to handle every new image. In strategy (a) every new image is registered to the last image which was added to the mosaic; in (b) every new image is registered to the contiguous image of the previous column.

As the size of the mosaic increases, distortions will usually appear in the mosaic due to accumulated errors in registration. It means that as the vehicle moves, the uncertainty (covariance) in the positional estimate of the vehicle increases with time. For this reason, Fleischer *et al.* proposed in [Fle97] a continuous optimal estimation

theory, so as to reduce the location error whenever the vehicle path crosses itself. The basic idea of this technique is to propagate back the error corrections around loops like the one illustrated in Figure 3.15. In this situation, the additional knowledge on the position of the vehicle can be propagated back through the image chain, improving the global placement of all the images of the mosaic.

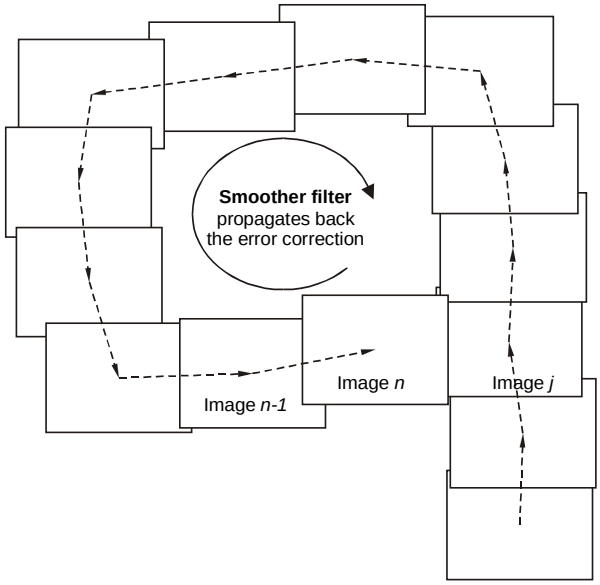


Figure 3.15. Arbitrary mosaic describing a rectangular trajectory in the XY plane while maintaining a constant heading. At the end of the trajectory the smoother filter [Fle97] minimizes the errors in the location of previous images of the mosaic.

By registering image *n* to image *j*, as well as image (*n*-1), an additional measurement of the global state of the n^{th} image is obtained. In addition, this new measurement is more accurate, since image *j* was consolidated earlier in the mosaic, and its location measurement had a lower variance. Therefore, drift can be corrected when the vehicle revisits a previously mapped zone. In this way, all the images placed between images *n* to *j* are relocated in the mosaic image by the filter, taking advantage of the extra positional information gained with the loop.

In [Fle96] the smoother filter was applied in a discrete fashion, assuming that the local displacements were constant between consecutive images. Unfortunately, this assumption was difficult to achieve in practice, as the acquisition of a new image before the vehicle has moved the desired displacement gave the system a higher

degree of robustness. Moreover, the derivation and implementation of the discrete algorithm when multiple loops of the vehicle are present is more difficult than its derivation in the continuous scenario. For this reason, the same authors later proposed a continuous version of its smoother filter [Fle97], preventing the system from experiencing the problems described above.

Unfortunately, the smoother filter used by MBARI/Stanford researchers assumes that the errors accumulate smoothly all over the loop. However, in practical situations the errors in building the mosaic are not distributed uniformly across the mosaic. On the contrary, at some points the error can be much larger than at other points, even though the line where the images are joined together at their edges has good visual registration. This effect is proved in [Sin98] (see Figure 4 in [Sin98]), where the researchers of WHOI together with the Johns Hopkins University analyze the quality of the visual mosaics by using extremely accurate (and expensive) navigation data. In [Sin98] the distortion across the mosaic is quantified by comparing the average distance of separation between the images that form the mosaic and the actual distance provided by the vehicle's accurate navigation system[1].

3.3 A comparative Classification

Several criteria can be applied in order to classify the previously described mosaicking techniques. A comparative scheme can be considered by categorizing the systems according to their motion detection technique. Using this principle, Table 2 proposes a general classification distinguishing between spatial (feature-based or featureless) and frequential methods.

The first row of the table identifies the different systems by giving the institution's name and the referred articles. The next two rows consider the necessity of correcting the distortions introduced by the lenses and the on-board lighting, respectively. It can be observed that while some of the works correct the lens

[1] This system is comprised of a conventional long-baseline acoustic navigation, a bottom-lock Doppler multibeam sonar, and a ring-laser gyroscope heading reference.

distortion, other systems have chosen a large focal-length camera to minimize this effect (reducing the field of view). Several techniques have been proposed to solve lighting inhomogeneities. Normally, scene radiance is assumed to change smoothly along with the image. Therefore, some authors [Neg98c] propose a radiometric model to compensate for variations in the scene illumination, while others [Rzh00] suggest the fitting of a surface to the gray-levels of the image, and then subtracting it from the original frame (*de-trending technique* in the table).

The following three rows of the table provide information about the nature of the motion performed by the vehicle, the assumed motion model and the technique that has been used to detect motion, respectively. Most of the systems consider the vehicle to be passively stable in pitch and roll; therefore those angles are not taken into account in order to estimate the vehicle motion. Furthermore, note the low incidence of frequential methods in the phase of motion estimation (only one of the systems that have been analyzed computes motion by means of the Fourier Transform). Feature-based techniques select image regions that maximize an interest criterion, such as the presence of zero-crossings of the Laplacian of the image [Mar95,Fle97], or a high-spatial gradient in both x and y directions [Gra00,Tru00a]. Some of those methods detect outliers among the selected features by means of robust techniques to discard false matches, such as X84 [Tru00a], LMedS [Odo99] or MEDSERE [Gra00].

The next two rows compare the image capture rate and the mosaic actualization rate, while the following row details whether the mosaicking system is able to work in real time as the vehicle is moving, or if the mosaic has to be constructed off-line. Next, the table specifies what is the main purpose of the mosaic: the construction of a visual map itself, or to serve as a navigation tool to estimate the motion of the submersible. Finally, the last two rows indicate which technique (if any) is applied to correct drift as the mosaic increases in size and how the input images are merged together to construct the mosaic.

Table 3.2. A summary of the analyzed mosaicking systems (ips: images per second; MR: most recent image; LR: less recent image; TM: temporal median; TA: temporal average)

	Feature-based				**Featureless**		**Frequential Techn.**
	Heriot-Watt University [Odo99] [Pla00] [Tru00a] [Tru00b]	**Instituto Sup. Tecnico (Lisboa)** + **Univ. Of Genova** [Gra98a] [San94] [Bra98c] [Mur94] [Gra00]	**Stanford University** + **MBARI** [Mar94b] [Mar95] [Mar94d] [Fle95] [Fle96] [Fle97] [Fle98] [Hus98]	**ENSTB + IFREMER** [Agu90], [Agu88]	**Underwater Vision and Imaging Laboratory** **Univ. Of Miami** [Xu97] [Str97] [Neg98d] [Neg98a] [Neg98e]	**Woods Hole Oceano-graphic Institution** [Eus00] [Sin98]	**University of New Hampshire** + **Heriot-Watt University** [Rzh00]
Distortion correction?	None	Radial distortion	Suggest the use of a narrow-field-of-view camera (20º)	No	Corrects the distortion due to refraction	None	None
Technique to compensate lighting problems	None	None	signum of LoG filter	Explicitly assume lighting conditions are constant.	Radiometric model	Adaptative histogram equalization Laplacian of Gaussian pyramid	Lighting Artifacts elimination through de-trending
Motion Model	Projective	- Translation and zoom - semi-rigid - affine - projective	4-parameter semi-rigid: Horizontal and vertical shift (t_x,t_y) scale factor (σ) rotation in the image plane (ψ)	2D translation, a trajectory is computed by adding all the displacements of the sequence.	Consider 3D translation and Yaw (t_x,t_y,t_z,ψ) (no pitch and roll)	- Affine - Projective	- Affine
Motion detection	Shi-Tomasi-Kanade tracker	Corner detection Correlation	Features with high spatial gradient detected through the zero crossings of the LoG operator	Apply a Sobel, Compute Hough (GHT), select the 5 best 80x80 windows	Direct Flow (Optical Flow)	Manual selection of matchings/ Automatic for simple motions (translation and rotation about the camera optical centre)	Fourier
Details of Computation of Registration parameters	1. Detect features of high spatial gradient; 2. Feature registration by minimizing frame regions in both images; 3. Use of robust regression techniques for *outlier* rejection (X84 in [Pla00,Tru00b], LMedS in [Odo99])	Use of robust regression techniques for *outlier* rejection before the computation of the 2D transformation (MEDSEDERE)	At frame rate (30 Hz) Live image is registered to the last image added to the mosaic. 1. Filtering (LoG)+Correlation. 2. Optimization technique to compute the 4 parameters.	Do not construct a mosaic, but find the trajectory followed by the submersible, choosing the first image as reference frame.	Uses small images to compute the registration 64×60 or 128×120	Levenberg Marquardt optimization procedure in [Eus00] Not detailed in [Sin98]	Translation determined from the phase shift theorem and by applying the Mellin transform to the Fourier transform to determine rotation & scaling factors.
Image Capture Rate	25 ips	25 ips	30 ips	5 ips (200 ms)	30 ips	Variable capture rate Processed offline	2-3 ips
Actualization of the mosaic	Fixed time	Fixed time	Fixed visual intervals. Only the images that incorporate some additional information are used to actualize the mosaic. (at a lower rate than 30 Hz)	None	Fixed-time Computationally more economical to update the mosaic every L frames	Not specified. Manual choice.	Fixed-time. Every image is taken into account (low frame-rate)
Real time?	Yes, reducing the number of ips	No	Yes, 30 Hz	No, CPU time for every image is 16 sec. (on a microVAX 3100)	Yes	No	Yes, the authors argue 2 fps is already real time.
Main application of the mosaic	Map Building	Map Building	Localization to aid navigation	Localization to aid navigation	Localization to aid navigation	Map Building	Map Building
Drift correction	Use a constant base frame	None	1. Consolidate images at fixed visual intervals (spare mosaic) 2. Looping trajectories	None	Consolidate images every L frames	None	None
Mosaic Actualization Strategy	Offline setup: MR /LR/TM/TA Real-time setup: MR/LR	MR /LR/TM	MR	None	LR	-	-

3.4 Conclusions

The main mosaicking techniques for aiding to autonomous underwater navigation have been reviewed in this Chapter, in order to point out the strengths and weaknesses of the different strategies. The comparative study could assist researchers in the decision on which techniques and solutions are the most appropriate to equip their vehicle with visual mosaicking capabilities.

One of the principal difficulties underwater vision systems have to face is that related to the lighting effects. The vehicle has to carry its own light source, producing non-uniform illumination, shadows and scattering effects. Several techniques have been proposed to compensate for these effects: LoG filtering, de-trending, radiometric models, etc. When using feature-based techniques, the Laplacian of Gaussian (LoG) operator has appeared as a widely used technique to locate features in front of lighting inhomogeneities. However, none of the proposed methods produces satisfactory results in the presence of backscatter or “marine snow”.

Image registration is a key step in the construction of visual mosaics. However, there is not a perfect methodology to recover the registration parameters between two images. Optical flow strategies are typically affected from the aperture problem, while feature methods do not suffer from this difficulty. However, feature-based correlation techniques have serious problems dealing with image rotations (yaw motion in mosaicking), and zooming effects. Most authors attenuate this problem by introducing the constraint of a high image capture rate. However, dense flow-based methods, though accurate, are computationally expensive and sensitive to local minima.

An evolution of flow methods can be found in the direct methods for motion estimation, which allow the estimation of 3D motion directly from spatio-temporal image gradient, without the need of any intermediate measure (such as: feature correspondences or the flow field). However, they suffer from the problems inherent in flow-based methods. Its accuracy also decreases as the apparent image motion is larger than one pixel. This inconvenience can be tackled by introducing a multi-resolution pyramidal scheme. In other respects, direct estimation methods are more

accurate than cross-correlation techniques in the estimation of motion over non-flat terrains. This is related to the fact that differences in depth create intersections in the image that do not correspond to a physical point of the scene. Nevertheless, it has been proved that when the texture is poor, the use of correlation-based algorithms provides better results than those obtained with differential techniques [Gia00].

Unfortunately, an open issue remains unsolved by present mosaicking systems: non-planar effects are extremely difficult to handle when the underwater environment is approximated by a plane. In this framework, the appearance of objects at different ranges produces a false perception of the world due to the perspective projection, as already stated in the literature [Mar95,Tiw96]. New strategies should be devised to find out when 3D effects are degrading the mosaic, leading to the construction of 3D underwater mosaics, that could be represented by a 3D virtual world, instead of a planar image.

According to [Sin98], there is no guarantee that, in practice, mismatches in the mosaic construction occur gradually and smoothly (as has been assumed in some looping-based mosaic correcting schemes [Fle95,Fle00]); on the contrary, sporadic impulse-type errors in the estimation of camera motion are more likely. For this reason, some authors (*i.e.* [Neg99]) uniquely actualize the mosaic within regions where no previous information exists. Then, when the vehicle moves to a zone where the image completely maps onto some part of the existing mosaic, this new information is only used for positioning correction. In our opinion, the looping strategy applied by Fleischer *et al.* [Fle97,Fle98] could be improved by introducing the error covariance of the correlation procedure into the system, instead of assuming the error propagates uniformly within the looping path.

Several researchers have already demonstrated real-time mosaicking with standard hardware. The advantages of real-time systems are twofold: firstly, they offer the possibility of providing navigational information while constructing the mosaic (Concurrent Mapping and Localization); secondly, the detection of gaps in the mosaic can be corrected within the same mission by revisiting the zone of interest. These mosaics should aid the robot navigation by acting as a world representation in which natural landmarks and areas of interest could be identified. These landmarks, together with the path planning algorithms, can be used to compute

the best paths towards the interest areas. However, other applications of visual mosaics do not require necessary on-line processing of the acquired images, such as the visualization of thermal vents, submerged structures, archeological sites or any observation performed by oceanic researchers in which the aim is to obtain a global perspective of the site of interest. In these cases, off-line construction of the mosaic should be enough providing that the mosaic covers the surveyed area without visual gaps.

References

[Agu88] F. Aguirre, J. M. Boucher and J. P. Hue, “Passive navigation of a submersible vehicle by image processign”, in *Proceedings of EUSIPCO*, pp. 963–966, 1988.

[Agu90] F. Aguirre, J. M. Boucher and J. J. Jacq, "Underwater navigation by video sequence analysis", in *Proceedings of the International Conference on Pattern Recognition*, pp. 537–539, 1990.

[Bal01] A. P. Balasuriya, and T. Ura, “Underwater Cable Following by Twin-Burger 2”, in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 920–926, 2001.

[Eus00] R. Eustice, H. Singh and J. Howland, “Image registration underwater for fluid flow measurements and mosaicking”, *Proceedings of the MTS/IEEE OCEANS Conference*, vol. 3, pp. 1529–1534, 2000.

[Fau86] O. D. Faugeras and G. Toscani, “The calibration problem for stereo”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 15–20, 1986.

[Fia96] M. Fiala and A. Basu, “Hardware design and implementation for underwater surface integration”, in *Proceedings of the IEEE/SICE/RSJ Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 815–822, 1996.

[Fle95] S. D. Fleischer, R. L. Marks, S. M. Rock and M. J. Lee, “Improved Real-Time Video Mosaicking of the Ocean Floor”, in *Proceedings of the MTS/IEEE OCEANS Conference*, pp. 1935–1944, 1995.

[Fle96] S. D. Fleischer, H. H. Wang, S. M. Rock and M. J. Lee, “Video Mosaicking Along Arbitrary Vehicle Paths”, in *Proceedings of the OES/IEEE Symposium on Autonomous Underwater Vehicle Technology*, pp. 293–299, 1996.

[Fle97] S. D. Fleischer, S. M. Rock and R. L. Burton, “Global Position Determination and Vehicle Path Estimation from a Vision Sensor for Real-Time Video Mosaicking and Navigation”, in *Proceedings of the MTS/IEEE OCEANS 97 Conference*, 1997.

[Fle98] S. D. Fleischer and S. M. Rock, “Experimental Validation of a Real-Time Vision Sensor and Navigation System for Intelligent Underwater Vehicles”, in *IEEE Conference on Intelligent Vehicles*, Stuttgart, Germany, October 1998.

[Gar00] R. Garcia and X. Cufi, “A Review on Mosaicking Systems for Underwater Applications”, Technical Report IIiA 00-16-RR, 2000.

[Gia00] A. Giachetti, "Matching techniques to compute image motion", *Image and Vision Computing*, no. 18, pp. 247–260, 2000.

[Gra00] N. Gracias and J. Santos-Victor, “Underwater Video Mosaics as Visual Navigation Maps”, *Computer Vision and Image Understanding*, vol. 79, no. 1, pp. 66–91, 2000.

[Gra97] N. Gracias and J. Santos-Victor, “Robust Estimation of the Fundamental Matrix and Stereo Correspondences”, in *Proceedings of the 5th International Symposium on Intelligent Robotic Systems*, also VisLab-TR 05/97, 1997.

[Gra98a] N. Gracias, “Application of robust estimation to computer vision: video mosaics and 3-D reconstruction”, Master thesis, ISR, 1998.

[Guo00] J. Guo, S.W. Cheng and J.Y. Yinn, “Underwater image mosaicing using maximum a posteriori image registration”, in *Proceedings of the International Symposium on Underwater Technology*, pp. 393–398, 2000.

[Ham86] F. R. Hampel, P. J. Rousseeuw, E. M. Ronchetti and W. A. Stahel, “Robust Statistics: the Approach Based on Influence Functions”, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, 1986.

[Har88] C.G. Harris and M.J. Stephens, “A combined corner and edge detector”, in *Proceedings of the Fourth Alvey Vision Conference*, Manchester, pp. 147–151, 1988.

[Hay86] R. Haywood, “Acquisition of a micro scale photographic survey using an autonomous submersible”, in *Proceedings of the OCEANS Conference*, vol. 5, pp. 1423–1426, 1986.

[Hor86] K. B. P. Horn, “Robot Vision”, Cambridge, Massachussets, MIT Press, 1986.

[How00] J.C. Howland and H. Singh, “Simulation of the deep sea mosaicking process”, in *Proceedings of the MTS/IEEE OCEANS Conference*, vol. 2, pp. 1353–1357, 2000.

[Hus98] A. Huster, S. D. Fleischer and S. M. Rock, “Demonstration of a vision-based dead-reckoning system for navigation of an underwater vehicle”, in *Proceedings of the OCEANS Conference*, Nice, France, pp. 185–189, September 1998.

[Lot01] J.-F. Lots, D. M. Lane, E. Trucco and F. Chaumette, “A 2-D Visual servoing for Underwater Vehicle Station Keeping”, in *Proceedings of the IEEE International Conference on Robotics and Automation*, Seoul, Korea, pp. 2767–2772, 2001.

[Mar80] D. Marr and E. Hildreth, “Theory of edge detection”, in *Proceedings of the Royal Society of London*, Series B, vol. 207, pp. 187–217, 1980.

[Mar94b] R. Marks, S. Rock and M. Lee, "Real-time video mosaicking of the ocean floor", in *Proceedings of IEEE Symposium on Autonomous Underwater Vehicle Technology*, 1994.

[Mar94d] R. Marks, M. Lee and S. Rock, "Using visual sensing for control of an underwater robotic vehicle", in *Proceedings of IARP Second Workshop on Mobile Robots for Subsea Environments*, Monterey, May 1994.

[Mar95] R. Marks, S. Rock and M. Lee, "Real-time video mosaicking of the ocean floor", *IEEE Journal of Oceanic Engineering*, vol. 20, no. 3, pp. 229–241, July, 1995.

[Neg93] S. Negahdaripour and C. H. Yu, "A generalized brightness change model for computing optical flow", in *Proceedings of the International Conference on Computer Vision*, Berlin, Germany, 1993.

[Neg98a] S. Negahdaripour, X. Xu and A. Khamene, "A vision system for real-time positioning, navigation and video mosaicing of sea floor imagery in the application of ROVs/AUVs", in *Proceedings of the 4th IEEE Workshop on Applications of Computer Vision*, pp. 248–249, 1998.

[Neg98c] S. Negahdaripour, "Revised definition of optical flow: integration of radiometric and geometric cues for dynamic scene analysis", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 9, pp. 961–979, 1998.

[Neg98d] S. Negahdaripour, X. Xu, A. Khamene and Z. Awan, "3D motion and depth estimation from sea-floor images for mosaic–based station–keeping and navigarion of ROVs/AUVs and high-resolution sea-floor mapping", in *Proceedings of the IEEE Workshop on Autonomous Underwater Robots*, pp. 191–200, 1998.

[Neg98e] S. Negahdaripour, X. Xu and A. Khamene, "Applications of direct 3D motion estimation for underwater machine vision systems", in *Proceedings of the OCEANS Conference*, vol. 1, pp. 51–55, 1998.

[Neg99] S. Negahdaripour, X. Xu, and L. Jin "Direct Estimation of Motion from Sea Floor Images for Automatic Station-Keeping of Submersible Platforms", *IEEE Journal of Oceanic Engineering*, vol. 24, no. 3, pp. 370–382, 1999.

[Odo99] F. Odone and A. Fusiello "Applications of 2D Image Registration, Research Memorandum", Research Memorandum RM/99/15, Heriot-Watt University, 1999.

[Olm00] A. Olmos, E. Trucco, K. Lebart and D. M. Lane, "Detecting Ripple Patterns in Mission Videos", in *Proceedings of the MTS/IEEE OCEANS*, Providence, Rhode Island, pp. 331–335, September 2000.

[Pla00] C. Plakas and E. Trucco, "Developing a real-time, robust, video tracker", in *Proceedings of the MTS/IEEE OCEANS Conference*, vol. 2, pp. 1345–1352, 2000.

[Rou87] P. Rousseeuw and A. Leroy, "Robust Regression and Outlier Detection", John Wiley & Sons, New York, 1987.

[Rzh00] Y. Rzhanov, L. Linnett and R. Forbes "Underwater Video Mosaicing for Seabed Mapping", in *Proceedings of the IEEE Conference on Image Processing*, 2000.

[Shi94] J. Shi and C. Tomasi "Good features to track", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600, 1994.

[Sin98] H. Singh, J. Howland, D. Yoerger and L. L. Whitcomb, "Quantitative photomosaicing of underwater imaging", in *Proceedings of the OCEANS Conference*, vol. 1, pp. 263–266, September 1998.

[Sna50] SNAME, The Society of Naval Architects and Marine Engineers, "Nomenclature for treating the motion of a Submerged Body Through a Fluid", Technical and Research Bulletin, no. 1–5, 1950.

[Sze94] R. Szeliski, "Image mosaicing for tele-reality applications", in *Proceedings of the IEEE Workshop on Applications of Computer Vision*, pp. 44–53, 1994.

[Tom91] C. Tomasi and T. Kanade, "Detection and tracking of point features", *Technical Report CMU-CS-91-132*, Carnegie Mellon University, Pittsburg, PA, 1991.

[Tom98] T. Tommasini, A. Fusiello, V. Roberto and E. Trucco, "Robust Feature Tracking in Underwater Video Sequences", in *Proceedings of the OCEANS Conference*, pp. 46–50, September 1998.

[Tru00a] E. Trucco, A. Doull, F. Odone, A. Fusiello and D.M. Lane, "Dynamic Video Mosaics and Augmented Reality for Subsea Inspection and Monitoring", in *Proceedings of the Oceanology International Conference*, pp. 297–306, 2000.

[Tru00b] E. Trucco, Y. R. Petillot, I. Tena Ruiz, K. Plakas and D. M. Lane, "Feature Tracking in Video and Sonar Subsea Sequences with Applications", *Computer Vision and Image Understanding*, vol. 79, no. 1, pp. 92–122, 2000.

[Tsa87] R.Y. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses", *IEEE Journal on Robotics and Automation*, vol. RA-3, pp. 323–344, August 1987.

[Wen92] J. Weng, P. Cohen and M. Herniou, "Camera calibration with distortion models and accuracy evaluation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 10, pp. 965–980, 1992.

[Xu00] X. Xu, "Vision-Based ROV System", Ph.D. Thesis, University of Miami, May, 2000.

[Xu97] X. Xu and S. Negahdaripour, "Vision-based motion sensing from underwater navigation and mosaicing of ocean floor images", in *Proceedings of the MTS/IEEE OCEANS Conference*, vol.2, pp. 1412–1417, 1997.

Chapter 4

Solving the correspondence problem

Detection of point correspondences is an essential part of the feature-based mosaicking strategies. Since our aim is to develop a mosaicking system based on the detection and matching of features, solving the correspondence problem in an accurate and reliable way is fundamental. Different alternatives to solve this problem by introducing texture analysis are considered in this Chapter. This is done in two phases: (a) comparing different texture operators individually, and (b) selecting those that best characterize the point/matching pair and using them together to obtain a more robust characterization. Various alternatives are studied to merge the information provided by the individual texture operators. Finally, the best approach in terms of robustness and efficiency is proposed.

4.1 Introduction

After analyzing the main mosaicking techniques described before, we aim to develop a feature-based mosaicking system. As can be derived from the previous Chapter, one of the key points of such a system is the accurate detection of correspondences. The correspondence problem can be stated as follows: given a set of features

detected in one image, find those same features in the next one, provided that the camera has undergone a small motion. Since there are, in general, several possibilities for the choice of the corresponding features in the second image, the correspondence problem is said to be *ambiguous* [Fau93], and therefore two questions emerge: which features are adequate? which constraints can be used to reduce this ambiguity? The selection of features is normally a complex task. Usually, points, lines or contours are selected. However, in the underwater environment, the already mentioned *lack of well-defined image features and contours* reduces this choice to points. The problem of feature detection will be described in Chapter 5, presenting an algorithm to detect the image points (called “interest points”) which are to be matched in the next image. In this Chapter we will focus on the detection of point correspondences in two consecutive images, without analyzing how the initial interest points are detected. The establishment of point correspondences will be done uniquely based on the information available in the images [Gar00,Gar01a]. Therefore, the second question, concerning the reduction of ambiguity from additional constraints will not be analyzed at this point. However, Chapter 5 will describe how the selection of a motion model serves as an additional constraint to reduce the ambiguity in the selection of matches, allowing the detection of false correspondences not describing the dominant motion [Gar01c].

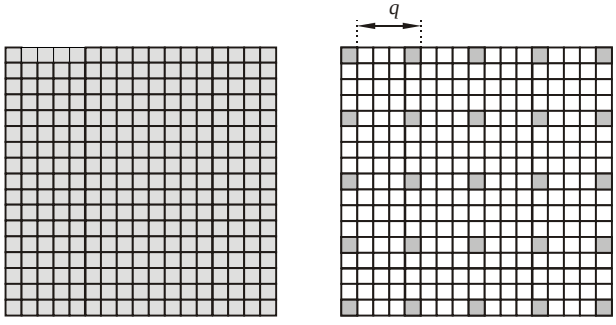
The proposed solution for the correspondence problem involves two phases. In the first phase, intensity-based correspondence analysis detects a set of possible matches of a given interest point. Then, the second phase applies texture analysis to this set of candidate matches, deciding which one among them corresponds to the correct match. Section 4.2 describes the first phase, while section 4.3 analyses the second one.

4.2 Intensity-based correspondence analysis

Within this Chapter we will assume that a set of points have been detected in the first image. These points, called “interest points”, determine the areas of the image which present a higher contrast. Starting from the interest points, this Chapter uniquely aims to find the corresponding points in the next image. In order to establish correspondences between the images, it is often assumed that corresponding pixels have a similar intensity value. However, identical intensity values occur in many

points of a given image. Therefore, several neighboring pixels in an image window are defined as one block. The assignment of corresponding points in both images is done using a *similarity measure* [Kle98], which is applied to the blocks associated with every point. As will be described later in this section, a classical correlation technique can be used as similarity measure [Gia00], obtaining the so-called "correlation score". This correlation is normally computed in intensity images. However, since our images are acquired by a color camera, we have found that in some cases the correlation produces better results in the blue band of the image. This fact is related to the variation of the optical properties of different water bodies depending on the interaction between the light and the aquatic environment [Fun72]. Given that the light suffers less absorption when it has a higher frequency, the blue component of the image provides higher contrast than the average of all frequencies, that is, the intensity component. Before correlation, our images are low-pass filtered with a Gaussian mask in order to reduce the inherent acquisition noise.

As the amount of interest points increases, the classic correlation approach is very time consuming. For this reason, we use a subsampled version of the correlation window. This means that if the correlation window has a size of $n \times n$ pixels, then only every q^{th} pixel of the window is taken into account, reducing the processed pixels to a $m \times m$ matrix, where $m = \left((n-1)/q\right)+1$. Figure 4.1 illustrates an example with n=17 and q=4. The accuracy of reducing the amount of data in the correlation window is practically the same as using the full window [Gia00]. This is due to the strong correlation in the gray level of neighboring pixels, producing smooth intensity variations, especially after the low-pass filtering.



(a) (b)

Figure 4.1. Data reduction of a correlation window of 17×17 pixels. (a) The whole set of 17 × 17 pixels is considered; (b) Only one pixel every four is taken into account to compute correlation, resulting in an effective size of 5 × 5 pixels.

Given an interest point **m** of the first image, with coordinates (x_1,y_1), the subsampled correlation window is centered at this point. Then, a search window is defined on the second image I', also centered at (x_1,y_1), as can be seen in Figure 4.2. Then the correlation operation is performed, computing a correlation score (CS) as *similarity measure* in the following way [Zha94]:

$$\mathrm{CS}(\mathbf{m},\mathbf{m}') = \frac{\sum_{i=-\alpha}^{\alpha}\sum_{j=-\alpha}^{\alpha}\left(I(x_1+i\cdot q, y_1+j\cdot q)-\overline{I(x_1,y_1)}\right)\cdot\left(I'(x_2+i\cdot q, y_2+j\cdot q)-\overline{I'(x_2,y_2)}\right)}{\alpha^2\sqrt{\sigma^2(I)\cdot\sigma^2(I')}} \tag{4.1}$$

where $\alpha = \left((n-1)\cdot q\right)/2$; I and I' are the first and second images, respectively, $\sigma^2(I)$ is the variance of the image computed in the correlation window (see equation (4.2)); and $\overline{I(x,y)}$ is the average of the correlation window in the image as shown in equation (4.3).

$$\sigma^2(I) = \frac{\sum_{i=-\alpha}^{\alpha}\sum_{j=-\alpha}^{\alpha} I(x+i\cdot q, y+j\cdot q)^2}{\alpha^2} - \overline{I(x,y)}^2 \tag{4.2}$$

$$\overline{I(x,y)} = \frac{\sum_{i=-\alpha}^{\alpha}\sum_{j=-\alpha}^{\alpha} I(x+i\cdot q, y+j\cdot q)}{\alpha^2} \tag{4.3}$$

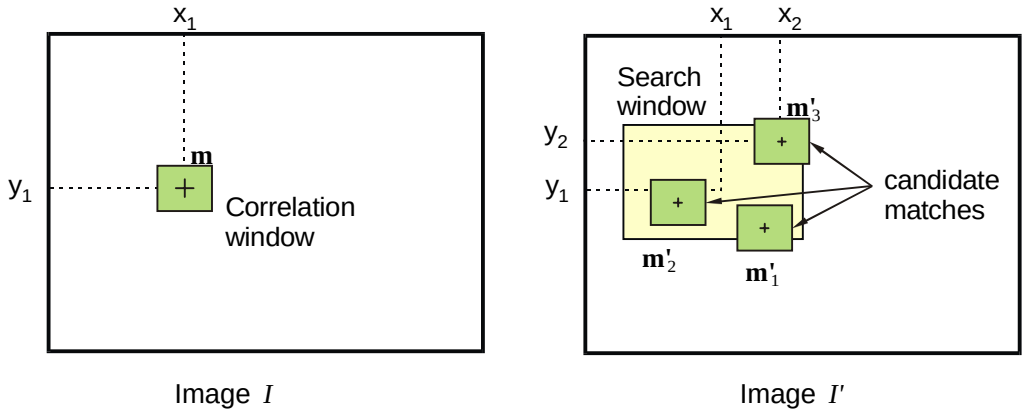


Figure 4.2: Typical situation where an interest point **m** of image *I* has several possible matches $\mathbf{m}'_i$ in image I'.

The correlation score of equation (4.1) returns a value in the interval $[-1,+1]$, where +1 means that the blocks associated to points **m** and $\mathbf{m}'$ are identical, and -1 indicates no similarity at all. Then, for a given interest point **m**, all the matchings $\mathbf{m}'_i$

with a similarity higher than 85% (*i.e.*, $CS \geq 0.7$) are added to a list of candidate correspondences of **m**. In this way, for each point in the first image, we thus have a set of *L* possible matches in the second image. The number *L* of candidate matches may be different from one interest point to another.

4.3 Point characterization from texture analysis

4.3.1 Introduction

Given an interest point **m** in *I*, the region-matching module provides a list $\{\mathbf{m}'_1, \mathbf{m}'_2, ..., \mathbf{m}'_L\}$ of *L* possible correspondences in I'. To decide which of these matches is the right one, the textural characteristics of these areas of image *I* are used as a matching vector to be correlated with the selected matches of the next image I'. An extensive study has been carried out in order to compare different texture operators. We have implemented and tested different configurations of the texture operators described in Chapter 2, *i.e.* co-occurrence matrix [Har73], energy filters [Law80], local binary patterns [Oja96], contrast features [Oja99] and symmetric covariances [Har95]. Some previous research of the *Computer Vision and Robotics Group* [Cuf98, Mar98, Fre00] has provided us with a larger set of texture operators. Among them, only non-structural operators have been considered since they are more adequate for describing underwater images, due to the unstructured nature of these images. These texture operators have also the advantage of being computationally efficient at the same time they provide good results in unstructured scenes. In this Chapter, the non-structural texture operators defined in Chapter 2 are compared individually to decide which among them are adequate to solve the problem of feature matching. Section 4.3.2 is devoted to this aspect.

Once the best textural parameters have been selected, it is then possible to merge them appropriately to obtain a more robust estimate. This second aspect is tackled in section 4.3.3. Different normalization approaches and comparison strategies will be considered to obtain the most accurate results by using the selected operators together. After this process, the texture analysis module will select the best candidate match as the correct correspondence.

4.3.2 Selection of the best texture operators

In this section the texture operators will be analyzed individually. Those texture parameters which demonstrate a good performance in detecting the correct correspondences from the list of candidate matches will be selected. The best texture operators will then be used in section 4.3.3 to provide more robustness to the system.

4.3.2.1 Characterization vectors

Now, the problem to solve can be stated as follows: we have detected a set of interest points in the first image I, and every interest point has several possible matchings in the next image I'. The aim of the texture analysis is to characterize all the points with their textural values, so that the best correspondence point-match could be established.

For a given texture operator, point characterization is performed by selecting a $m \times m$ window around the point. The texture operator is then computed in this neighborhood. In this way, a *characterization vector* $\mathbf{v}$ of size $m \times m$ is obtained. Typical values of m are 7, 9 or 11. Therefore, if m=7 is selected, the characterization vector stores 49 values. Every value v_i is the result of applying the considered texture operator to that pixel (and its neighborhood). We should bear in mind that the value given by a texture operator in a point involves not only a measurement on this point, but in the whole region where this point is located, as stated in Chapter 2, where we saw that texture is a property of regions. Taking this into account, it is easy to see that there is a high degree of redundancy when characterizing the point considering the texture of every pixel in its neighborhood. However, later in this Chapter we will see that subsampling strategies can be applied to the $m \times m$ window, saving computing effort at the cost of reducing redundancy.

4.3.2.2 Similarity Measures

Once the interest points and matches have been characterized, it is necessary to find a means to compare them, in order to decide which is the best correspondence for a

given interest point. The values that form the characterization of a point are stored in a characterization vector **v**.

Four different approaches have been considered to compare the textural characterization vector of an interest point **m**: $\mathbf{v} = [v_1, v_2, ..., v_N]$ and that of the matching $\mathbf{m}' : (\mathbf{v}' = [v'_1, v'_2, ..., v'_N])$. These similarity measures are the following:

(i) **Mean distance**. Compute the average (μ) of the values stored in both characterization vectors, and then substract them to obtain a similarity measurement $d_{n,m}$.

$$d_{n,m} = \left|\mu_n - \mu'_{n,m}\right| \qquad \text{with } \mu = \frac{\sum_{i=1}^{N} v_i}{N} \tag{4.4}$$

where v_i is the i^{th} element of the characterization vector , μ_n and $\mu'_{n,m}$ are the mean of the point *n* in image *I* and the m^{th} candidate matching in image I', respectively; and *N* is the size of the vector (*N*= *m*×*m*).

(ii) **Standard deviation**. Compute the standard deviation (σ) of both *characterization vectors,* and then compare them by subtraction.

$$d_{n,m} = \left|\sigma_n - \sigma'_{n,m}\right|, \qquad \text{with } \sigma = +\sqrt{\sum_{i=1}^{N} \frac{(v_i - \mu)^2}{N}} \tag{4.5}$$

where σ_n and $\sigma'_{n,m}$ are the standard deviation of the interest point *n* and the m^{th} candidate matching, respectively.

(iii) **Mean of point-to-point distances**. Compute an average of the point-to-point distances $\delta(\mathbf{v}, \mathbf{v}')$ between every component of the characterization vectors of the point $(\mathbf{v} = [v_1, v_2, ..., v_N])$ and the matching $(\mathbf{v}' = [v'_1, v'_2, ..., v'_N])$.

$$\delta(\mathbf{v}, \mathbf{v}') = \frac{\sum_{i=1}^{N} \left|v_i - v'_i\right|}{N} \tag{4.6}$$

(iv) **Euclidean distance**. Compute the Euclidean distance between both characterization vectors.

$$d(\mathbf{v},\mathbf{v}') = \sqrt{\sum_{i=1}^{N}\left(v_i - v_i'\right)^2} \tag{4.7}$$

It should be noted that the first two similarity measures (mean and standard deviation) fuse the spatial information available in the characterization vector into a single value. On the contrary, similarity measures (iii) and (iv) analyze two characterization vectors by comparing the components which belong to the same spatial zone. In this way, spatial information is taken into account when analyzing similarity. For this reason we expect a better "a priori" behavior of (iii) and (iv).

The next section shows the experimental results obtained through testing the different configurations of the textural operators described in Chapter 2. The similarity measures detailed above are used to test the operators.

4.3.2.3 Results

4.3.2.3.1 Experimental methodology

We aim now to compare the different texture operators described in Chapter 2. Every operator has different parameters which can be combined. Consider, for instance, the energy filter: first a mask to pre-filter the image is selected (among the nine 3×3 masks and as many again of the 5×5 masks), then one of the statistical measures is computed (mean, standard deviation positive mean or negative mean) in the selected neighborhood (3×3 or 5×5). This results in a combinatory explosion if all the combinations are to be considered. Taking this into account, it is necessary to devise an automatic strategy to test the texture operators. With this purpose, the following methodology has been applied:

1. First, a representative set of pairs of consecutive underwater images has been selected in order to test the accuracy of the correspondences. These images try to be a good representation of the different conditions that the submersible could find in a real sea mission.
2. For every pair of images, the detector of interest points has been applied to the first image of the sequence (as will be described in Chapter 5).

3. The *intensity-based correspondence analysis* methodology described in Section 4.2 detected a set of candidate matches in the second image. For every interest point in the first image, all the matches with a correlation score higher than 0.7 (*i.e.* a similarity measure of 85%) have been stored in the list of candidate matches.
4. A specific software is then used to obtain the set of correct matchings through the supervision of a human expert. The expert selects the correct matching of every interest point among the candidate correspondences. If the correct match does not appear in the list of candidate matches, it is manually included in this list. Since this operation may be error-prone, five different human experts have performed this operation.

By considering the last point, three types of matchings have been defined:

(a) **correct**, when the correspondence detected after texture analysis is the same as the one selected by the human experts. A tolerance of one pixel is allowed.

(b) **incorrect**, the correspondence does not coincide with the right one, including the one-pixel tolerance.

(c) **nearly correct**, which means that the obtained correspondence does not coincide with the one selected by a minimum of three of the human experts, but at least one of the human experts has selected this match (considering the tolerance).

At this point, a list of interest points and their corresponding matches is available. It should be noted that the methodology described above is uniquely used to obtain a perfectly matched set of image points for every sequence. Information regarding which are the *correct* matches (and the *nearly correct* ones) will be used "a posteriori" in the next section only to check if a given texture operator has selected the correct match within the candidate list.

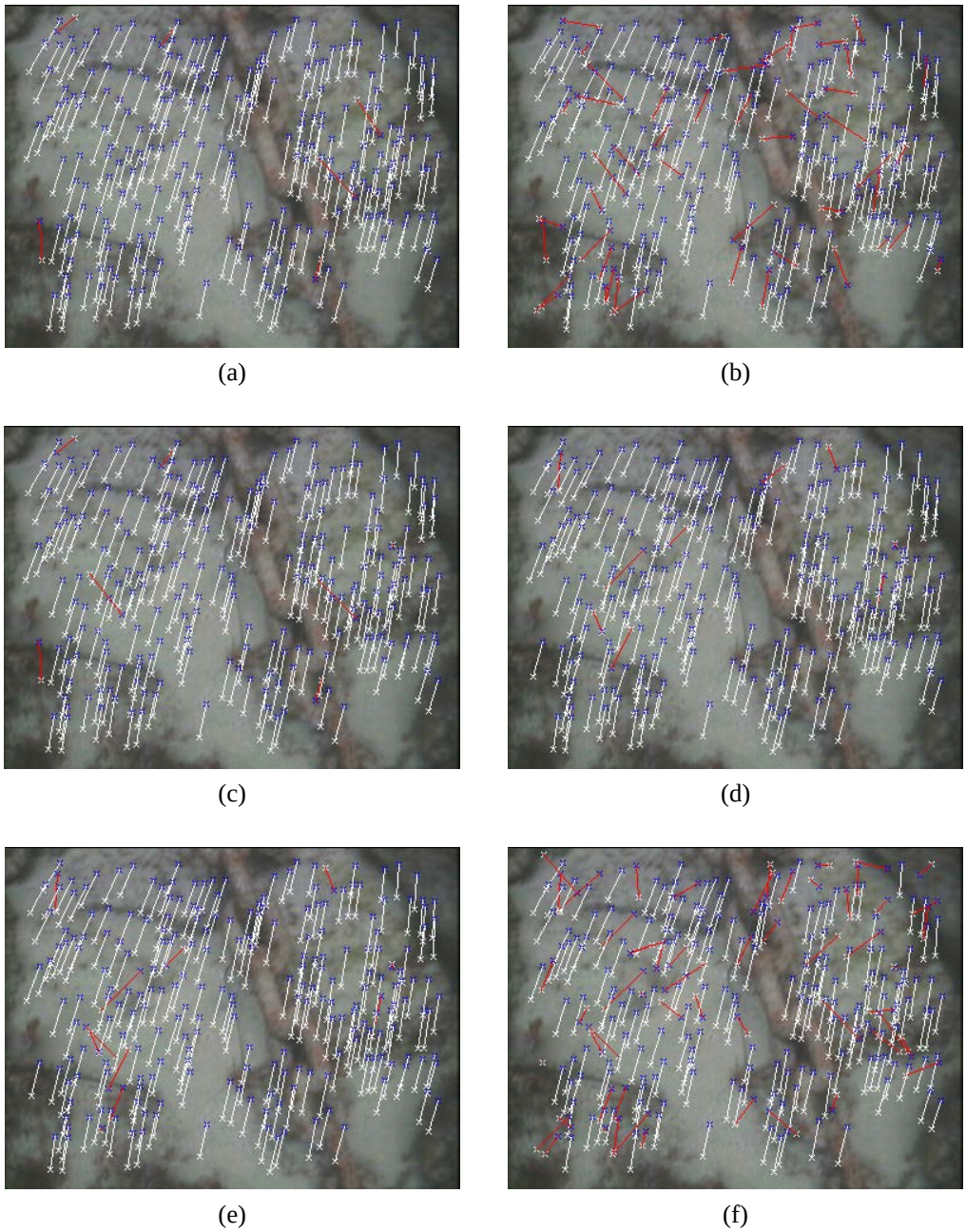
4.3.2.3.2 Testing the individual texture operators

A total of 9 underwater sequences have been used to test the individual texture operators. A comparison among the high amount of texture operators can be performed automatically, since the right correspondences are known beforehand. Figure 4.3 illustrates the algorithm which has been used to evaluate the individual

texture operators. After executing this algorithm, considering all the texture operators described in Chapter 2, a raking of the best texture operators can be extracted. A very reduced set of representative results is shown in Figures 4.4 and 4.5. The interest points are drawn in blue, and the matches in white. Every pair of corresponding points is connected by means of a vector. These vectors are drawn in white if the selected correspondence is the *correct* one, and in red otherwise. From the images in Figure 4.4 it can be seen that the selection of an adequate *similarity measure* to compare the characterization vectors of the interest points and the candidate matches is very important. *Euclidean distance* (*ED*) and *Mean of point-to-point distances* (*MPTPD*) provide better results than *mean* or *standard deviation* as similarity measures. This result is directly related to the fact that ED and MPTPD keep the information regarding the spatial distribution of the elements of the characterization vector. In practically all cases, ED has performed a bit better than MPTPD, thought with a small difference. Thereby, Euclidean distance is the similarity measure which has been considered as the proper choice for comparing two characterization vectors.

```
ALGORITHM Test_Individual_texture_operators
   Load List of Interest_Points in I
   Load List of Candidate_Matches in I'
   Load List of Right_Matches in I'
   FOR every_Texture_Op DO
      FOR i = 1 TO Num_Int_Points DO
         v_i = Char.Vector of the Interest_Point
         min = INFINITY
         FOR j = 1 TO Num_Candidate_Matches DO
            v'_{i,j} = Char.Vector of the Candidate_Match
            SM = Compute Similarity( v_i , v'_{i,j} )
            IF SM < min THEN
               CM = Set j as Correct_Match
            ENDIF
         ENDFOR
         IF CM = Right_Match(i) THEN
            Update count of Wrong_Matches
         ENDIF
      ENDFOR
      Return num. Wrong_Matches for this operator
   ENDFOR
ENDALGORITHM
```

Figure 4.3. Algorithm to automatically test all the texture operators. It is executed for the nine test sequences.



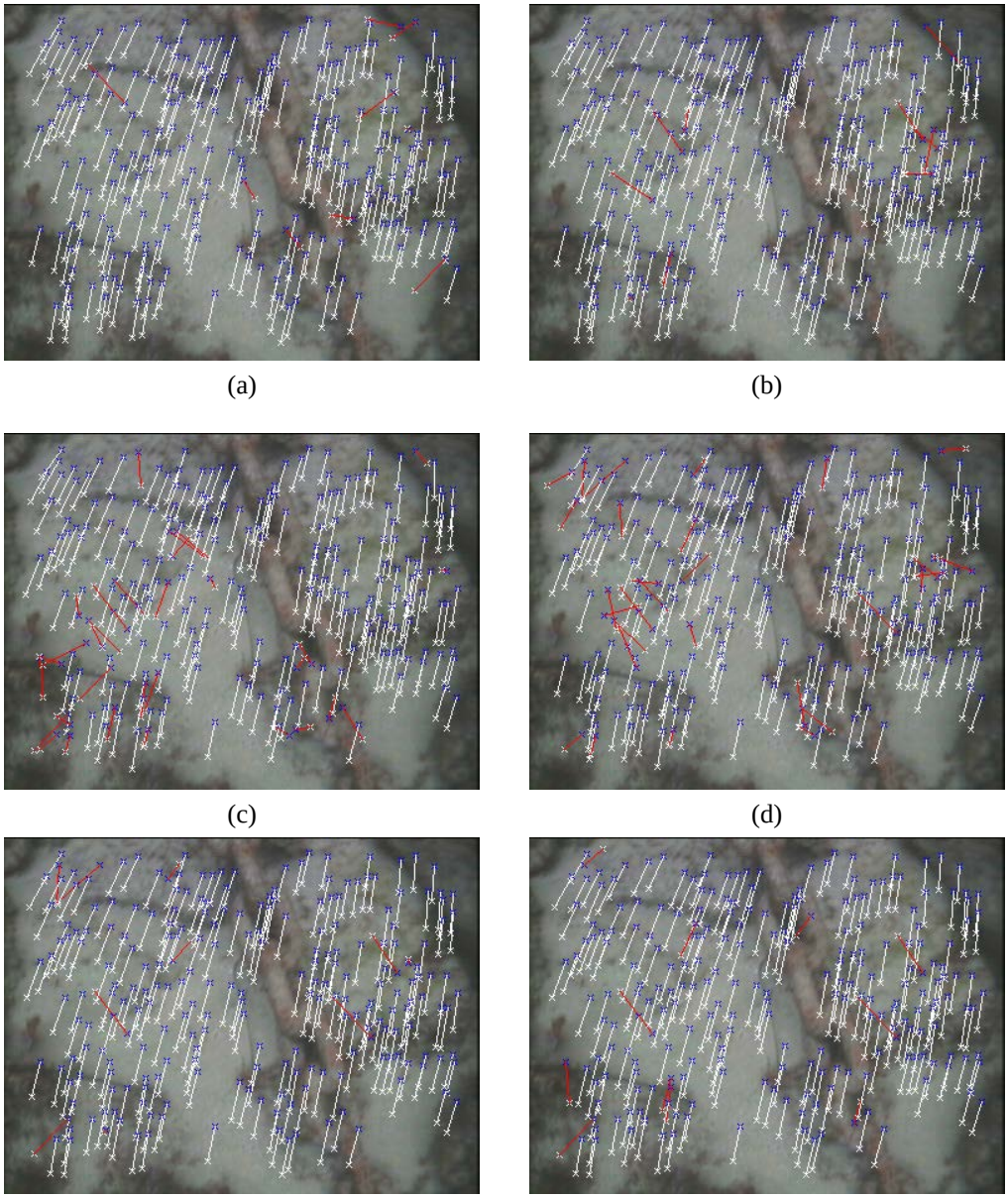
(a) (b)

(c) (d)

(e) (f)

Figure 4.4. Detection of correspondences in two consecutive images from 232 points; Characterization neighborhood (*Char*.); Similarity measure (*S*:); amount of wrong correspondences (*W*).

(a) Energy L3L3 with 3×3 Std. dev.; ***Char***. 7×7; ***S***: Euclidean distance; ***W***: 6.
(b) Energy L3L3 with 3×3 Std. dev.; ***Char***. 7×7; ***S***: Mean; ***W***: 49.
(c) Energy L3L3 with 3×3 Std. dev.; ***Char***. 7×7; ***S***: mean of point-to-point distances; ***W***: 7.
(d) Energy E3E3 with 3×3 pos. mean; ***Char***. 7×7; ***S***. Euclidean distance; ***W***: 9.
(e) Energy E3E3 with 3×3 pos. mean; ***Char***. 7×7; ***S***: mean of point-to-point distances; ***W***: 11.
(f) Energy E3E3 with 3×3 pos. mean; ***Char***. 7×7; ***S***: mean. ***W***: 51.



(a) (b)

(c) (d)

(e) (f)

Figure 4.5. Detection of correspondences in two consecutive images from 232 points; Characterization neighborhood (*Char*.); Similarity measure (*S*:); amount of wrong correspondences (*W*).

(a) Energy L5S5 with 3×3 pos. mean ***Char***. 7×7; ***S***: Euclidean distance; ***W***: 9.

(b) Energy E5S5 with 3×3 neg. mean; ***Char***. 7×7; ***S***: Euclidean distance; ***W***: 9.

(c) 16×16 co-occurrence matrix (d=1 and $\theta = 0^{o}$) with 3×3 Uniformity; ***Char***. 7×7; ***S***: Euclidean distance; ***W***: 31.

(d) 3×3 LBP; ***Char***. 7×7; ***S***: Euclidean distance; ***W***: 29.

(e) 3×3 Contrast; ***Char***. 7×7; ***S***: Euclidean distance; ***W***: 9.

(f) 5×5 Contrast; ***Char***. 7×7; ***S***: Euclidean distance. ***W***: 11.

Due to the high volume of data involved, the results global results of the experimental validation of the individual texture operators have been summarized in Figures 4.6–4.9 and their corresponding Tables 4.1–4.3. First, the algorithm of Figure 4.3 has been applied to the nine selected underwater sequences. A characterization vector of size 7×7 has been used. This strategy provides a percentage of incorrect matches for every configuration of all the texture operators. Then, this percentage can be averaged for all the sequences. The resulting list can be ordered in such a way that the operators which detect less incorrect matches are listed first. Figure 4.6 shows the percentage of incorrect correspondences obtained when applying the algorithm to test the texture operators to the nine sequences. A list of the best 20 texture operators is provided in Table 4.1. The average error varies between a 5% and a 7%, approximately.

The same operation has been performed to compare again the whole set of texture operators, but considering a characterization vector of size 9×9 and 11×11. Figures 4.7 and 4.8 and their corresponding tables show these results, respectively. It can be observed from these data that the percentage of incorrect matches decreases as the size of the characterization vector increases. For the sake of space, only the error averages are shown in Figures 4.6–4.9. However, it should be remarked that when the processed images undergo a rotation, accuracy decreases as size of the characterization vectors increases. This reason, together with the loss of computing efficiency as the neighborhood increases, has limited the size of the characterization vectors that have been considered in this dissertation.

Once the individual performance of the best texture operators has been studied, our aim is now to select some of them to be used jointly, providing a more measure of correspondences. Unfortunately, for a given sequence, some of the best operators detect the same wrong matches. For this reason, the criteria to select the most efficient textural operators has to be applied taking into account that the incorrect matches should be different from one selected operator to another. When two operators provide quite similar results, our selection has taken into account their computational cost.

Table 4.1. List of the best texture operators of Figure 4.6. The characterization vector is constructed considering a neighborhood of size 7×7.

Texture operator **7×7**	***Selected*** **(✓)**	***Percentage of incorrect correspondences***
Energy L3L3 Negative mean 3×3		4.8241
Energy L3L3 Negative mean 5×5		4.8241
Energy L3L3 Negative mean 7×7		4.8241
Energy L3L3 Negative mean 15×15		4.8241
Energy L7S7 Positive mean 3×3		5.1446
Energy L7S7 Absolute mean 3×3		5.1610
Energy L3E3 Absolute mean 3×3		5.2548
Energy E7S7 Negative mean 3×3		5.3064
Energy E7S7 Positive mean 3×3		5.4337
Energy L3L3 Standard deviation 3×3	✓	5.4374
Energy L5S5 Positive mean 3×3	✓	5.5240
Energy L7S7 Positive mean 5×5		5.9288
Energy L5S5 Negative mean 3×3		6.0531
Energy L7S7 Negative mean 3×3		6.0741
Energy E3E3 Positive mean 3×3	✓	6.4932
Contrast 3×3	✓	6.5987
Energy E5S5 Negative mean 3×3	✓	6.6539
Energy L5S5 Absolute mean 3×3		6.7935
Energy L3L3 Standard deviation 5×5		6.9031
Energy L3E3 Negative mean 3×3		6.9193
Energy E3E3 Negative mean 3×3	✓	7.1013

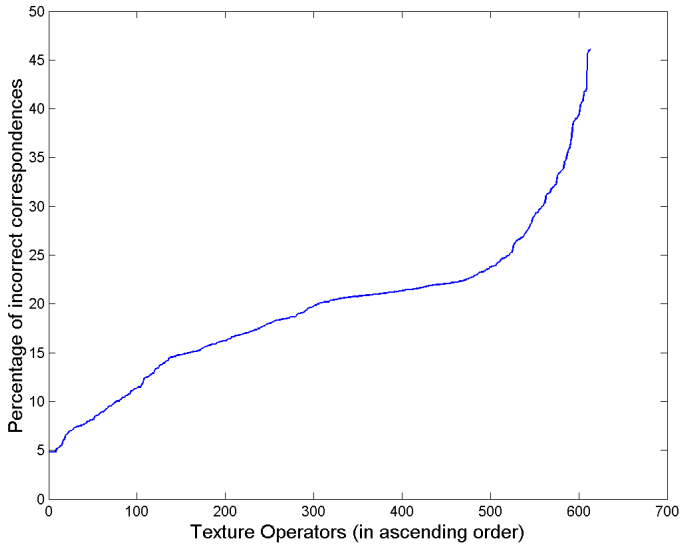


Figure 4.6. Percentage of incorrect correspondences obtained by means of a 7×7 characterization vector. The texture operators are ordered according to the obtained error percentage.

Table 4.2. List of the best texture operators of Figure 4.7. The characterization vector is constructed considering a neighborhood of size 9×9.

***Texture operator* 9×9**	***Selected* (✓)**	***Percentage of incorrect correspondences***
Energy L5S5 Positive mean 3×3	✓	2.7873
Energy L7S7 Positive mean 3×3		3.1107
Energy E7S7 Positive mean 3×3		3.4642
Energy E3E3 Positive mean 3×3	✓	3.4898
Energy E7S7 Negative mean 3×3		3.7088
Energy L7S7 Absolute mean 3×3		4.0367
Energy L3E3 Positive mean 3×3		4.0510
Energy L3L3 Standard deviation 3×3	✓	4.1372
Energy L7S7 Positive mean 5×5		4.1923
Energy E5S5 Negative mean 3×3	✓	4.2704
Energy L7S7 Negative mean 3×3		4.2965
Energy L3E3 Absolute mean 3×3		4.3553
Energy L3E3 Negative mean 3×3		4.4483
Energy E3E3 Absolute mean 3×3	✓	4.5453
Energy L5S5 Negative mean 3×3		4.5933
Energy L5S5 Positive mean 5×5		4.6348
Energy L5S5 Absolute mean 3×3		4.7818
Energy L3L3 Standard deviation 5×5		4.9313
Energy E7L7 Negative mean 5×5		5.0457
Energy L3L3 Negative mean 3×3		5.0473
Energy L3L3 Negative mean 5×5		5.0473

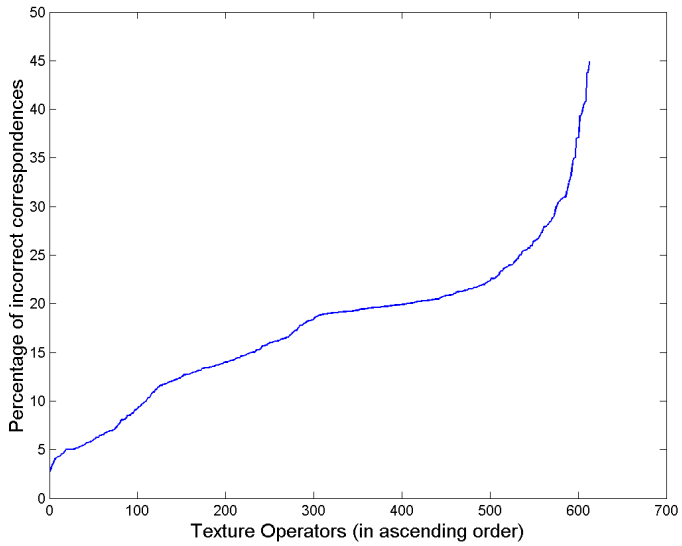


Figure 4.7. Percentage of incorrect correspondences obtained by means of a 9×9 characterization vector. The texture operators are ordered according to the obtained error percentage.

Table 4.3. List of the best texture operators of Figure 4.8. The characterization vector is constructed considering a neighborhood of size 11×11.

***Texture operator* 11×11**	***Selected* (✓)**	***Percentage of incorrect correspondences***
Energy L7S7 Positive mean 3×3		1.7163
Energy L5S5 Positive mean 3×3	✓	1.9151
Energy L3E3 Absolute mean 3×3		2.0697
Energy L3L3 Standard deviation 3×3	✓	2.1015
Energy E7S7 Positive mean 3×3		2.1402
Energy E7S7 Negative mean 3×3		2.1555
Energy L7S7 Negative mean 3×3		2.1998
Energy L7S7 Absolute mean 3×3		2.2532
Energy E3E3 Absolute mean 3×3	✓	2.2882
Contrast 3×3	✓	2.3391
Energy E3E3 Positive mean 3×3	✓	2.4046
Energy L3E3 Positive mean 3×3		2.4096
Energy L5S5 Negative mean 3×3		2.4426
Energy L7S7 Positive mean 5×5		2.4752
Energy E7L7 Absolute mean 3×3		2.5637
Energy L3L3 Standard deviation 5×5		2.6966
Energy E7L7 Negative mean 5×5		2.6993
Energy E5S5 Negative mean 3×3	✓	2.8360
Energy E5L5 Absolute mean 3×3		2.8686
Energy E5L5 Standard deviation 3×3	✓	2.8933
Energy E3L3 Negative mean 5×5		2.9724

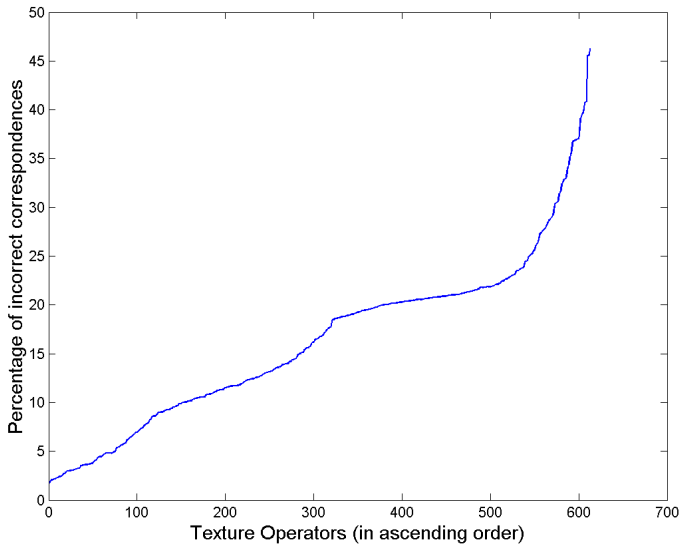


Figure 4.8. Percentage of incorrect correspondences obtained by means of an 11×11 characterization vector. The texture operators are ordered according to the obtained error percentage.

From this information, it can be seen that some of the texture operators have provided quite good results. Nine of them have been selected among the whole set, since their performance to solve the correspondence problem for the set of test images has been clearly better than the others. These operators are the following:

- Operator 1 – L3L3 Energy with 3×3 Standard Deviation
- Operator 2 – E3E3 Energy with 3×3 Positive Mean
- Operator 3 – E3E3 Energy with 3×3 Negative Mean
- Operator 4 – L5S5 Energy with 3×3 Positive Mean
- Operator 5 – E5L5 Energy with 3×3 Standard Deviation
- Operator 6 – E5S5 Energy with 3×3 Negative Mean
- Operator 7 – Contrast feature considering a 3×3 neighborhood
- Operator 8 – Contrast feature considering a 5×5 neighborhood
- Operator 9 – Contrast feature considering a 7×7 neighborhood

The last two operators (8 and 9) do not appear among the very best ones (although they are in the best 50 out of a total of 613 combinations). However, they have been included here since present a completely different distribution of incorrect matches when compared to the first 7 texture operators. Nevertheless, when real time performance is required, the selection can be limited to the first 7.

Hereafter, some results are shown as a sample. Table 4.4 shows the results that have been obtained for the nine selected operators, characterizing the interest points and candidate matches with a neighborhood of size 7×7. The first column of the Table indicates which images have been used and the amount of interest points which have to be matched. The next columns show the results for every one of the selected operators, and the last column indicates which results are obtained considering the correspondence with the highest correlation score through intensity-based correspondence analysis, as described in section 4.2. If only one number appears in a cell of the table, it indicates the amount of incorrect matches. When two numbers appear, separated with a “hyphen”, the first digit denotes the amount of *incorrect* matches, and the second one the number of *nearly correct* correspondences.

Table 4.4. Amount of incorrect matches obtained by the selected operators. The characterization vector is constructed considering a neighborhood of size 7×7.

Images	*No. pairs*	*Op.*1	*Op.*2	*Op.*3	*Op.*4	*Op.*5	*Op.*6	*Op.*7	*Op.*8	*Op.*9	*Corr.*
Seq. 1	232	6	8-1	9-3	9	19-4	7-2	9	9-2	10-1	6-2
Seq. 2	184	4	5	4	1	11	3	4	8	11	3
Seq. 3	169	11-2	15-1	8	8	15-1	12-1	10	13-2	15	6-1
Seq. 4	158	5	6	9	4	12	4	8	9	9	4
Seq. 5	294	3	17-2	18-1	13-2	10	21-3	17-2	13	12-3	2-1
Seq. 6	123	8	3	5	7	4	5	3-1	10	8-1	4
Seq. 7	301	8-2	11-1	17-2	14-2	28-4	9-1	18-2	11-3	7-5	11-4
Seq. 8	126	3-1	4	3	1	2	3	3-1	5-1	3-1	2-2
Seq. 9	229	2	5	4	6	7-1	5	3	8	14	1

The intensity-based correspondence analysis has been performed considering a 21×21 correlation window, and subsampling factor $q = 2$ (see equation 4.1, page 90 for details). These parameters have been chosen after an experimental validation, since they provide the best ratio between the correct matches and the highest correlation score among the candidate correspondences. From Table 4.4 it can be seen that the amount of wrong matches detected by some texture operators for a given sequence is better than those found by the correlation procedure. Nevertheless, none of the texture parameters is able to provide a better average result than correlation.

Tables 4.4 and 4.5 show the same results when characterizing the points with a neighborhood of sizes 9×9 and 11×11, respectively. Larger neighborhoods have not been considered due to the computational burden. From these tables, it can be observed that the average incorrect matches detected by the texture operators is worst than considering correlation, though not for a big difference. Initially, we expected to find some texture operator which would perform better than correlation. However, it was confirmed from these results, that the *incorrect* and *nearly correct* matches where quite different in the selected operators. For this reason, the idea of using several texture operators which form different incorrect matches seemed quite attractive.

Table 4.5. Amount of incorrect matches obtained by the selected operators. A characterization vector of size 9×9 has been considered.

Images	*No. pairs*	*Op.*1	*Op.*2	*Op.*3	*Op.*4	*Op.*5	*Op.*6	*Op.*7	*Op.*8	*Op.*9	*Corr.*
Seq. 1	232	3-2	5-1	7-1	5	14-3	4-3	6	7	3-3	6-2
Seq. 2	184	3	4	3	1	7	1	3	5	7	3
Seq. 3	169	6-1	8	12-1	6	12-2	11-1	10	15-1	17-1	6-1
Seq. 4	158	5	3	5	1	6	2	7	7	9	4
Seq. 5	294	2-1	10	13	3	7	14-2	16-1	10	11-1	2-1
Seq. 6	123	7	2	5	4	2	3	3	7	9-1	4
Seq. 7	301	7-2	10-1	6-4	8-3	14-2	8	13-2	5	5-3	11-4
Seq. 8	126	3-1	2	1-1	0	0	1	3	4	3	2-2
Seq. 9	229	2	3-1	1	2	3	2	5	5	10	1

Table 4.6. Amount of incorrect matches obtained by the selected operators considering a 11×11 characterization vector.

Images	*No. pairs*	*Op.*1	*Op.*2	*Op.*3	*Op.*4	*Op.*5	*Op.*6	*Op.*7	*Op.*8	*Op.*9	*Corr.*
Seq. 1	232	1-4	1	6	3	8-2	1	2	3-2	1-3	6-2
Seq. 2	184	2	2	1	1	3	1	2	3	3	3
Seq. 3	169	5	8	6	4	8-1	11	9	10-1	13-2	6-1
Seq. 4	158	0	3	3	1	5	0-1	3	3	6	4
Seq. 5	294	0-1	7	3	4	2	11-1	7	6	5	2-1
Seq. 6	123	4	1	2	1	2	1	4	7	9	4
Seq. 7	301	2	7	5-3	5	12-3	3	10-2	3	5	11-4
Seq. 8	126	2	2	1	1	0	2	1	2	3	2-2
Seq. 9	229	1	1	2	2	4	2	2	5	4	1

Since the false matches detected by the selected texture operators are quite different in every case, the next section is devoted to find the means of combining these nine operators in an adequate manner to obtain a better correspondence measure. Moreover, the experimental validation has also proved that the *Euclidean distance* is the best similarity measure to compare two characterization vectors $\mathbf{v}$ and $\mathbf{v}'$. Therefore, uniquely this measure will be used below.

4.3.3 Characterization with multiple operators

Although the selected texture operators have provided quite good results, it is possible to enhance the global result by using these texture operators together. In this way, once the best individual operators are selected, they have to be combined adequately to obtain a more robust operator, which should be able to efficiently solve the correspondence problem. Thus, the difficulty is to find an adequate method to combine these texture operators. Two different approaches have been considered for merging the results obtained with these texture operators:

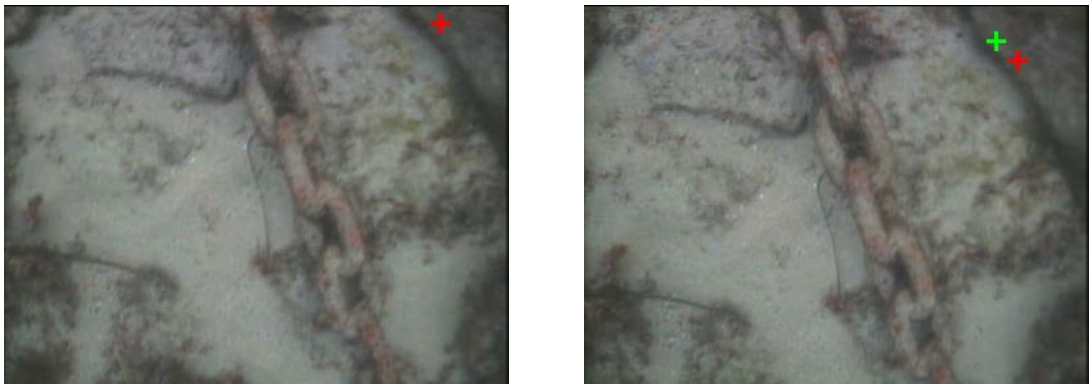
a) Sum of distances. Compute the distance between the texture vector of the interest point **v**, and the vector of the matching $\mathbf{v}'$ for every texture operator, as explained above for the case of a single texture operator. Then, the sum of distances of the different texture operators is computed and a single value is obtained.

b) Vote Assignation. Assign votes to every candidate correspondence $\mathbf{m}'_i$ depending on how good this match is for every texture operator. Thus, for *L* possible correspondences, the best match would receive *L* votes, *L*–1 the second best, and so on. Once all the texture operators have voted, the selected match will be the one which has obtained more votes.

4.3.3.1 Sum of distances

This alternative consists of applying the nine selected operators to a given interest point and all its candidate matches. Then, for every texture operator the Euclidean distance can be computed and the average of the resulting distances can be computed for every candidate match. The correspondence with the smaller average distance can be taken as the right one. However, this approach has a serious limitation: the dynamic range of every operator may be different, and the true correspondence could be affected by a high distance in one of the texture operators, being discarded in front of some other match. We will illustrate this problem with an example. Consider for instance the pair of images shown in Figure 4.9. Given the interest point **m** = (332,14) detected in the first image, two candidate matches detected through correlation will be taken into consideration: $\mathbf{m}'_1 = (330, 42)$ and $\mathbf{m}'_2 = (313, 27)$,

being $\mathbf{m}'_1$ the right one. It should be mentioned that correlation selects $\mathbf{m}'_2$ as the correct one.



(a) (b)

Figure 4.9. Two consecutive images of one of the sequences. The interest point is drawn in image (a), while matches $\mathbf{m}'_1$ and $\mathbf{m}'_2$ are drawn in image (b) in red and green, respectively.

The interest point and the two matches will be characterized by means of the 9 selected texture operators and considering a 7×7 neighborhood. When the point **m** is characterized, the distribution of values illustrated in Figure 4.9 is obtained. As can be seen in this Figure, every texture operator generates a distribution of values with a different dynamic range. The Figure shows a darker point in the areas where more points are concentrated, and the point is lighter when few values coincide in that area. Then, both candidate correspondences are also characterized, and the Euclidean distance between the interest point **m** and every match is computed for each and every texture operator. Table 4.7 shows the resulting values for every texture operator. It is clear from both the Table and Figure 4.10 how the operator 5 (*E5L5 Energy with 3×3 Standard Deviation*) generates very high values in the characterization vector, giving rise also to a high Euclidean distance measurement. Although this operator provides good results in other cases, in this case it does not identify which one is the right correspondence, while all the other texture operators do. It is obvious from this example that a normalization or reparametrization strategy is needed to prevent one operator to have more weight than all the others.

Table 4.7. Euclidean distance computed without normalizing the characterization vector. The incorrect match has a smaller Euclidean distance when no normalization is performed.

	Correct match: $\mathbf{m}'_1 = (330, 42)$	*Wrong match*: $\mathbf{m}'_2 = (313, 27)$
Text.op.1	54.506882	67.260689
Text.op.2	4.690416	10.392304
Text.op.3	5.000000	7.745967
Text.op.4	95.456795	74.752930
Text.op.5	347.692383	296.005066
Text.op.6	11.090536	23.302361
Text.op.7	11.618950	13.638182
Text.op.8	10.246951	15.165751
Text.op.9	9.000000	17.058722
Total	**549.3029**	**525.3219**

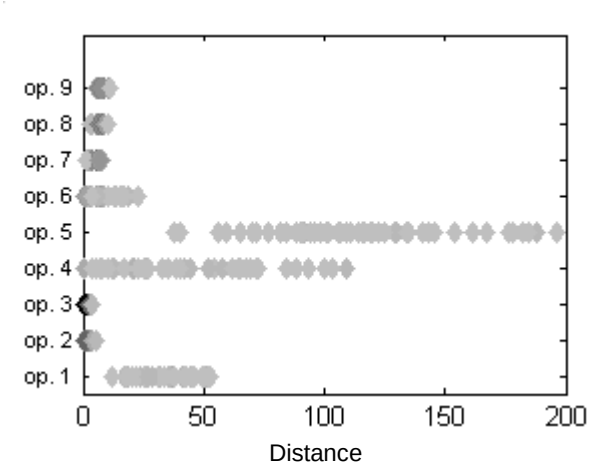


Figure 4.10. Distribution of the values that characterize the point **m**, as defined by every texture operator.

4.3.3.1.1 Data Standarization

Five normalization/reparametrization methods have been proposed to adjust (standarize) the values of the characterization vectors:

1. **Independent normalization**. Independent normalization of every characterization vector. The normalization process takes into account the mean (μ)

and standard deviation (σ) of the vector. The normalized vector is obtained from:

$$u_i = \frac{v_i - \mu}{\sigma} \quad \forall v_i,\ i = 1..N \tag{4.8}$$

where $\mathbf{u} = [u_1, u_2, ..., u_N]$ is the new normalized characterization vector of the point **m**: $\mathbf{v} = [v_1, v_2, ..., v_N]$; and the mean and standard deviation are computed as follows:

$$\mu = \frac{\sum_{i=1}^{N} v_i}{N} \quad \text{and} \quad \sigma = +\sqrt{\sum_{i=1}^{N} \frac{(v_i - \mu)^2}{N}} \tag{4.9}$$

2. **Joint normalization**. Normalization of every vector taking into account the characterization vectors of the interest point **v** and the all candidate matches $\mathbf{v}'_j$ to compute the global mean (μ_G) and standard deviation (σ_G):

$$\mu_G = \frac{\sum_{i=1}^{N} v_i}{N} + \sum_{j=1}^{Y} \frac{\sum_{i=1}^{N} v'_{ji}}{N} \tag{4.10}$$

$$\sigma_G = +\sqrt{\sum_{i=1}^{N} \frac{(v_i - \mu_G)^2}{N} + \sum_{j=1}^{L} \sum_{i=1}^{N} \frac{(v'_{ji} - \mu_G)^2}{N}} \tag{4.11}$$

where *L* is the number of candidate correspondences of the interest point **m**.
Thus, the normalized characterization vector $\mathbf{u} = [u_1, u_2, ..., u_N]$ of the interest point is:

$$u_i = \frac{v_i - \mu}{\sigma} \quad \forall v_i,\ i = 1..N \tag{4.12}$$

And the normalized vector of the j^{th} correspondence $\mathbf{u}'_j = [u'_{j1}, u'_{j2}, ..., u'_{jN}]$ is obtained from:

$$u'_{ji} = \frac{v'_{ji} - \mu}{\sigma} \quad \forall v'_{ji},\ j = 1..L,\ i = 1..N \tag{4.13}$$

3. **Independent reparametrization**. Independent reparametrization of every characterization vector by taking into account the minimum and maximum value of the vector.

   Consider $\mathbf{v} = [v_1, v_2, ..., v_N]$ the characterization vector of point **m**, and $\mathbf{u} = [u_1, u_2, ..., u_N]$ the same vector once reparametrizated. Then:

$$u_i = \frac{v_i - v_{\min}}{v_{\max} - v_{\min}} \qquad \forall v_i,\ i = 1..N \tag{4.14}$$

   where $v_{\min}$ is the minimum value of vector **v**, and $v_{\max}$ is the maximum value of **v**, *i.e.*, $v_{\min} = \min(\mathbf{v})$ and $v_{\max} = \max(\mathbf{v})$.

4. **Reparametrization considering all the candidate correspondences**. Reparametrization of every characterization vector considering the minimum and maximum value of the vectors of the interest point and all the candidate matchings. Then:

   $v_{\min} = \min\left(\min(\mathbf{v}), \min(\mathbf{v}'_1), ..., \min(\mathbf{v}'_L)\right)$

   $v_{\max} = \max\left(\max(\mathbf{v}), \max(\mathbf{v}'_1), ..., \max(\mathbf{v}'_L)\right)$

$$u_i = \frac{v_i - v_{\min}}{v_{\max} - v_{\min}} \qquad \forall v_i,\ i = 1..N \tag{4.15}$$

$$u'_{ji} = \frac{v'_{ji} - \mu}{\sigma} \qquad \forall v'_{ji},\ j = 1..L,\ i = 1..N \tag{4.16}$$

5. **Reparametrization considering the theoretical values**. Reparametrization of every characterization vector considering the theoretical minimum and maximum values which every texture operator can achieve.

   Consider the characterization vector **v** of a point **m**, with a given texture operator, and **u** the normalized version of **v**. Then:

   $v_{\min}$ = minimum value that the texture operator can achieve

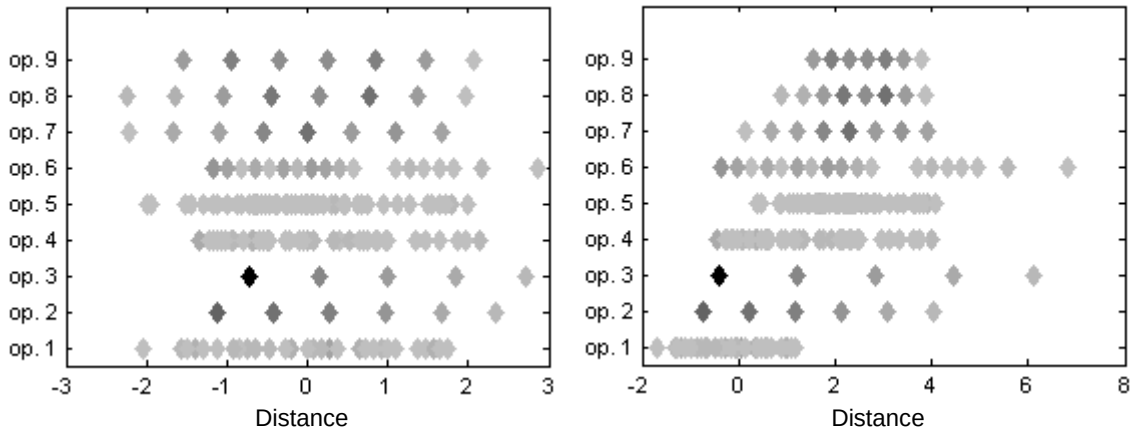
   $v_{\max}$ = maximum value which can be achieved by the texture operator

   Again, the reparametrizated vector $\mathbf{u} = [u_1, u_2, ..., u_N]$ can be computed from:

$$u_i = \frac{v_i - v_{\min}}{v_{\max} - v_{\min}} \qquad \forall v_i,\ i = 1..N \tag{4.17}$$

It can be seen from the equations above that a normalization implies a centering of the data set (substracting the *mean*), and a grouping of the further data (dividing by the *standard deviation*). On the contrary, reparametrization simply adjusts the dynamic range of the data, without taking into account its dispersion.

Retaking the example of the interest point **m** = (332,14) and the two candidate matches $\mathbf{m}'_1 = (330, 42)$ and $\mathbf{m}'_2 = (313, 27)$, Figures 4.11–4.13 show the distribution of values obtained from every texture operator, after applying some of the previously described normalization/reparametrization methods.



(a) (b)

Figure 4.11. Distribution of the normalized values that characterize the point **m**.
(a) *Independent normalization.* The normalization has been performed individually for every texture vector.
(b) *Joint normalization*. Normalization taking into account the *standard deviation* and *mean* which is obtained from the point and all its candidate matches.

The distribution of values shown in Figure 4.11(a) illustrates how *independent normalization* produces a smaller dispersion of the data after normalization. However, when the point **m** is normalized taking into account all the possible correspondences (15 in this example), the range of values that form the characterization vector shows again a higher dispersion (see Figure 4.11(b)). In this second case, the characterization vectors are normalized taking the mean and

standard deviation of both the 15 matches and the interest point. Tables 4.8 and 4.9 show the resulting Euclidean distances in both cases.

Table 4.8. *Independent normalization.* Euclidean distance normalizing separately the characterization vector.

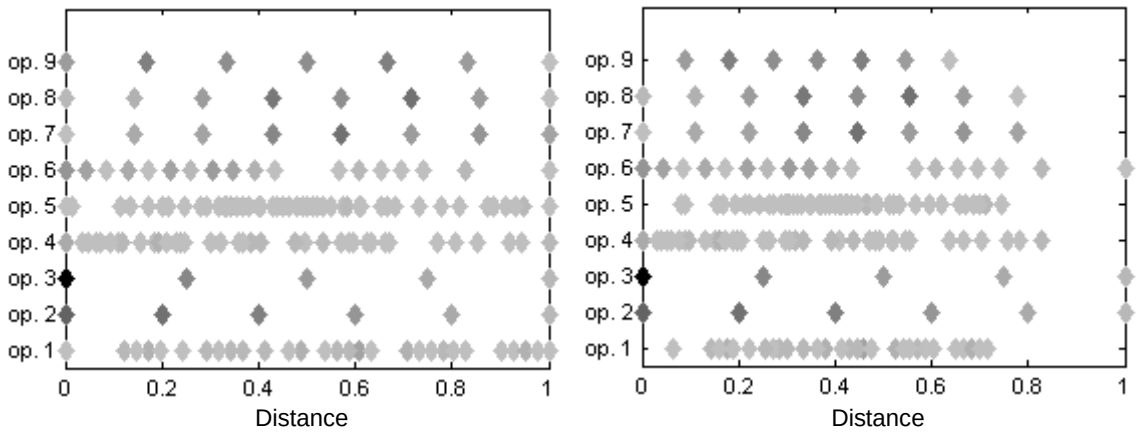
	Correct match: $\mathbf{m}'_1 = (330, 42)$	*Wrong match*: $\mathbf{m}'_2 = (313, 27)$
Text.op.1	4.978095	5.221354
Text.op.2	3.294620	7.787081
Text.op.3	4.401453	7.071034
Text.op.4	2.873065	2.369292
Text.op.5	6.694543	7.667400
Text.op.6	1.944944	4.235385
Text.op.7	6.352748	7.417632
Text.op.8	5.705553	7.520783
Text.op.9	5.189359	8.151355
Total	**41.4343**	**79.8900**

Table 4.9. *Joint normalization*. Euclidean distance normalizing the characterization vector with the standard deviation computed from the interest point and all the candidate matches.

	Correct match: $\mathbf{m}'_1 = (330, 42)$	*Wrong match*: $\mathbf{m}'_2 = (313, 27)$
Text.op.1	3.769718	4.651776
Text.op.2	4.456625	9.874305
Text.op.3	8.131666	12.597523
Text.op.4	3.882510	3.040422
Text.op.5	8.114323	6.908063
Text.op.6	3.457730	7.265045
Text.op.7	6.282333	7.374125
Text.op.8	4.371123	6.469375
Text.op.9	3.395176	6.435265
Total	**45.8612**	**64.6158**

It can be seen from the Tables above that in both normalization cases the right match obtains a smaller Euclidean distance. This means that the characterization vector of $\mathbf{m}'_1$ is more similar to that of $\mathbf{m}$, giving rise to a proper selection of the correct correspondence.

Figure 4.12 shows the distribution of values of **m** performing a reparametrization of every characterization vector. It can be observed from Figure 4.12(a) how the values obtained by the texture operators after *independent reparametrization* have been scaled so that all of them work in the same range.



(a) (b)

Figure 4.12. Distribution of values obtained from reparametrization.
(a) *Independent reparametrization*. Reparam. is performed individually for every texture vector.
(b) *Reparametrization considering the point and all the candidate correspondences*.

Finally, Figure 4.13 illustrates the reparametrization of every characterization vector considering the minimum and maximum theoretical value of the texture operator. In this approach, the resulting values are very small, since for a given example, it is very difficult to achieve the maximum and minimum theoretical values which can be obtained by every texture operator.

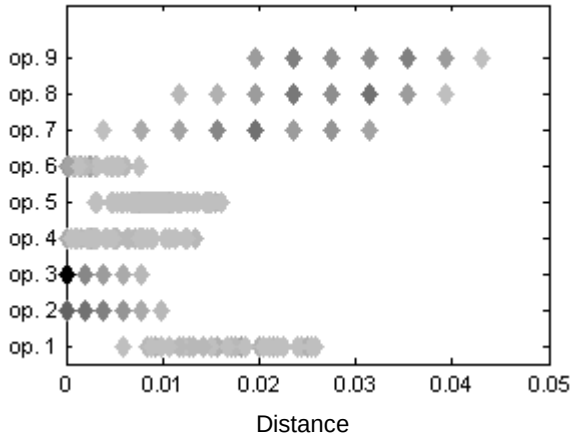


Figure 4.13. *Reparametrization considering the theoretical values.* The minimum and maximum theoretical value of every texture operator are considered.

Once the values of the characterization vectors have been normalized/reparametrizated by means of one of the methods described above, the sum of the distances using the normalized vectors can be performed, since all the texture operators work in the same dynamic range.

4.3.3.1.2 Weighting the distances

However, there is still a point that should be considered. Above we have made a difference between *normalize* and *reparametrize*. In the second case, when we *reparametrize* a characterization vector, the distribution of the values of the vector is not taken into consideration. For this reason, it is convenient to weight the distances (be it Euclidean or average of distances) with the standard deviation. The higher the variability of the components of the vector, the higher the deviation and, therefore, distance should be weighted to be less relevant if it has to be added with other distances. But, this deviation can be computed in two different ways:

(a) Taking into account both the components of the characterization vectors of the interest point **u** and the candidate correspondence $\mathbf{u}'$. The standard deviation is computed as follows:

$$\sigma = +\sqrt{\sum_{i=1}^{N}\frac{(u_i-\mu)^2}{N}+\sum_{i=1}^{N}\frac{(u_i'-\mu)^2}{N}}\text{, with } \mu=\frac{\sum_{i=1}^{N}u_i}{N}+\frac{\sum_{j=1}^{N}u_i'}{N} \tag{4.18}$$

Then, the distance *d* measuring the difference between both vectors is weighted with the standard deviation σ. This distance can be the Euclidean distance or an average of distances, as detailed in Section 4.4.2.2.

$$d_{weigh}^{(1)} = d/\sigma \tag{4.19}$$

(b) Taking into consideration both the components of the characterization vectors of the interest point **u** and all the candidate correspondences $\mathbf{u}'_j$ found by the region correlation algorithm. The standard deviation is computed as follows:

$$\sigma = \sqrt{\sum_{i=1}^{N}\frac{(u_i-\mu)^2}{N}+\sum_{i=1}^{N}\sum_{j=1}^{L}\frac{(u_{ji}'-\mu)^2}{N}}\text{, with } \mu=\frac{\sum_{i=1}^{N}u_i}{N}+\sum_{i=1}^{N}\frac{\sum_{j=1}^{L}u_{ji}'}{N} \tag{4.20}$$

Obtaining the weighted distance $d_{weigh}^{(2)}$:

$$d_{weigh}^{(2)} = d/\sigma \tag{4.21}$$

4.3.3.2 Vote Assignation

A second approach to jointly use the nine selected texture operators consists in assign votes to every candidate correspondence $\mathbf{m}'_i$ depending on how good this match is for every texture operator. This idea is quite intuitive if we consider the example of Table 4.7 (in page 108), where the different characterization vectors are considered without data standarization. In that example, only 2 of the texture parameters have a smaller Euclidean distance in the case of the wrong match, while the other 7 parameters consider the correct correspondence as the good one (because its

Euclidean distance is lower). Therefore, the result would have been 7 votes against 2 (the "good match" wins). Instead of giving a unique vote to the best match, better results are obtained as follows: for L possible correspondences, the best match would receive L votes, second best L–1, and the worst match would receive just 1 vote. Once all the texture operators have voted, the selected match will be the one which has obtained more votes. With this second strategy, a match which is considered as the second best for all the texture operators has a good chance to be elected, while it did not have any if only one vote is emitted by every operator.

Vote assignation can also be used after some *data standarization* process has occurred. Therefore, all the normalization/reparametrization methods presented in the previous sections should be considered to test the results of this approach.

4.3.3.3 Subsampling the characterization vector

In order to speed up computation, one of the important issues to be addressed is checking whether the subsampling of the *characterization vector* deteriorates texture characterization. We expect the strong correlation between the measured texture in neighboring points to allow a subsampling of the considered characterization window without affecting too much the results. To perform the tests, parameter m has been set to 7, 9 and 11, which typically represent a good trade-off between accuracy and computational demands. Higher values of m have been avoided since the hypothesis of negligible deformations of the pattern inside the window fails as the characterization window increases its size. The considered subsampling procedures are illustrated in Figure 4.14 for every case.

Table 4.10 shows the amount of values that are taken into account to characterize the texture of a point, depending on the size of the selected neighborhood and the subsampling strategy.

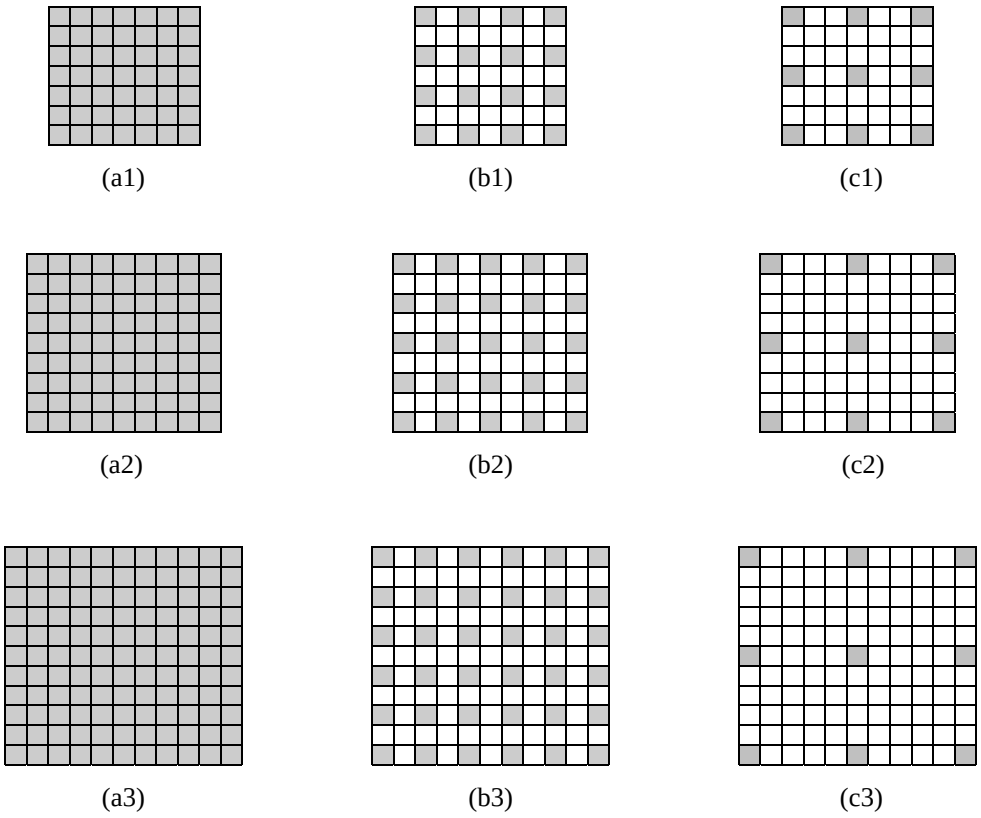


Figure 4.14. Proposed data subsampling for the characterization vector. In the first row, a neighborhood of 7 × 7 pixels is considered; the second row takes a 9 × 9 neighborhood; in the last row 11 × 11 pixels are considered. Different subsamplings are selected for every column (a) – (c).

Table 4.10 Size of the characterization vector, depending on the neighborhood considered (7×7, 9×9 or 11×11), and the applied subsampling (a), (b) or (c) as shown in Figure 4.14.

		Neighborhood		
		7 × 7	**9 × 9**	**11 × 11**
Subsampling	**(a)**	49	81	121
	(b)	16	25	36
	(c)	9	9	9

4.3.3.4 Results

Some of the results obtained for the previously described strategies are presented below. However, a more detailed set of results can be found in *Appendix A*. These results try to answer the main unknowns we aim to solve: (1) what is the ideal size of the window used to characterize the region centered at an interest point? (2) which is the best normalization/reparametrization method which should be applied? and (3) is it worth sub-sampling the characterization window in terms of efficiency and accuracy?

First, we will look at the size of the characterization window. It was stated before that the window should be big enough to keep a sufficient amount of information inside the window, but it should also be small enough to keep negligible deformations of the pattern inside the window. For this reason the tests have been limited to square neighborhoods of size 7, 9 and 11. Both alternatives (*sum of distances* and *vote assignation*) have been tested to use simultaneously several texture operators. Figures 4.15 and 4.16 confirm our initial guess: the higher the characterization window, the better the results. The results of obtained through texture analysis are compared to those obtained from correlation.

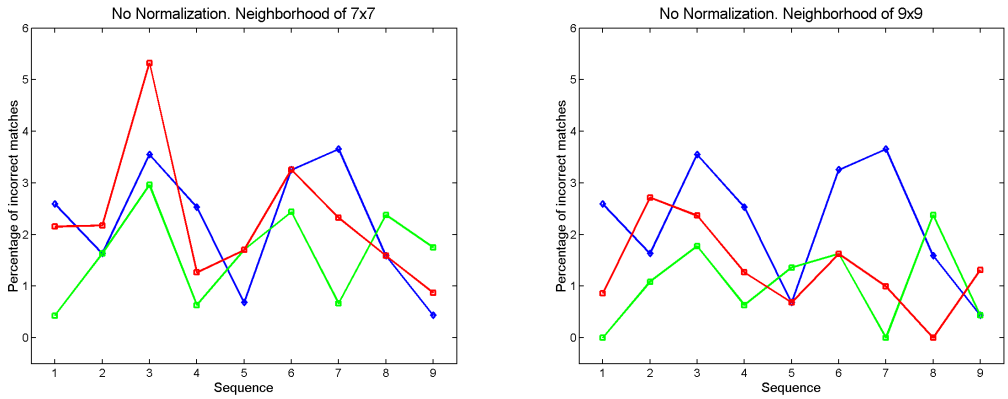


Figure 4.15. Percentage of incorrect matches without normalization obtained by considering a neighborhood of 7×7 (left) and 9×9 (right). The *sum of Euclidean distances* is drawn in red, and *vote assignation* in green. The percentage obtained by *Correlation* is shown in blue.

It can be observed from Figure 4.16 (left) that subsampling the characterization window obtains worst results than using the whole window.

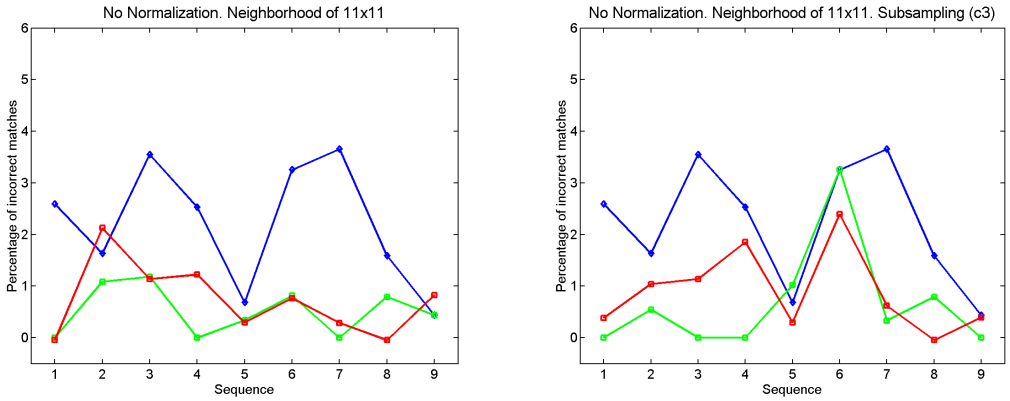


Figure 4.16. Percentage of incorrect matches without normalization obtained by considering a neighborhood of 11×11 and subsampling (c3) of Figure 4.14: *sum of Euclidean distances* (red) and *vote assignation* (green). The percentage obtained by *Correlation* is shown in blue.

In the case where neither normalization nor reparametrization is applied, the *voting strategy* (green) provides better results than the *sum of Euclidean distances* (red). This indicates that vote assignation is less sensitive to the different dynamic range of the measures.

Next, both alternatives are considered when standarizing the data with the strategies described in the previous section. These strategies are only shown for the case of an 11×11 neighborhood. *Appendix A* shows the performance of the proposed standarization strategies with further detail, including smaller characterization windows. Figures 4.17 and 4.18 show the percentage of incorrect correspondences obtained through *independent normalization* and *independent reparametrization*. As it was discussed in the section devoted to data standarization, results are improved after standarization, as expected. After data standarization, sum of Euclidean distances produces better results than vote's assignation. The reparametrization strategy, which takes into account the minimum and maximum value of the characterization vector, results in a slightly better percentage of correct matches than normalization. It should be noted that subsampling with strategy (c3) of Figure 4.14 does not deteriorate to a large extent the accuracy, while drastically reducing the complexity of a factor 14 (from 11×11 to 3×3).

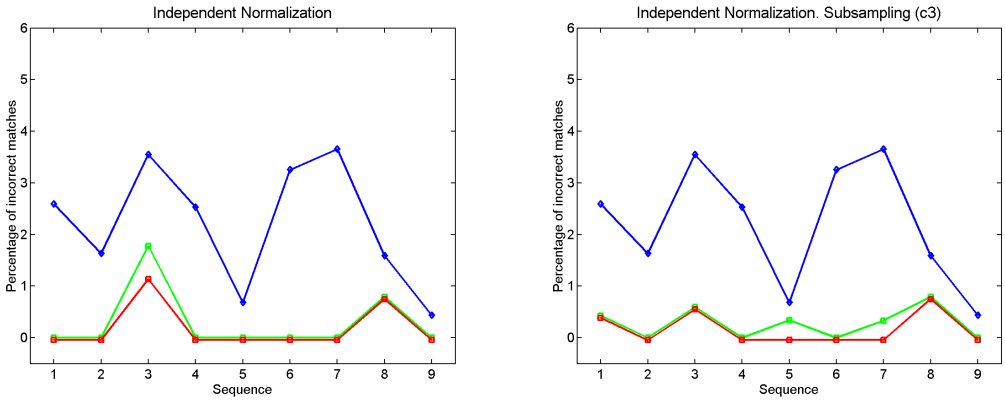


Figure 4.17. Percentage of incorrect matches (*independent normalization*), considering a neighborhood of 11×11 and subsampling (c3): *sum of Euclidean distances* (red) and *vote assignation* (green). The percentage obtained by *Correlation* is shown in blue.

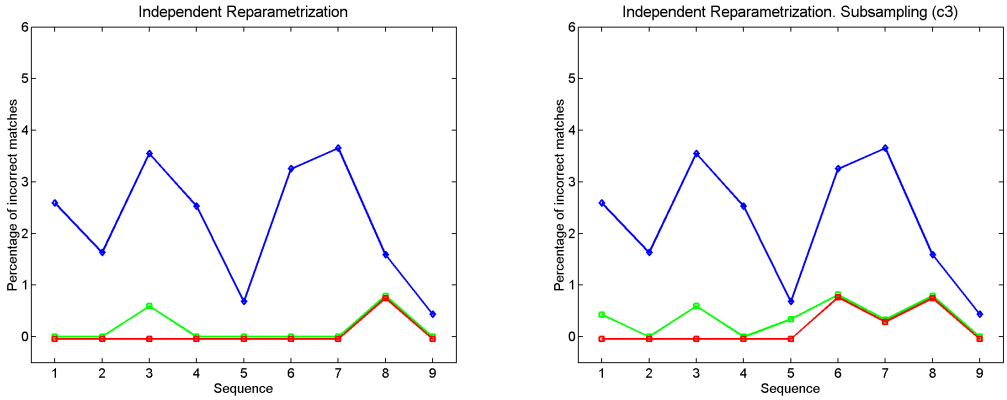


Figure 4.18. Percentage of incorrect matches (*independent reparametrization*), considering a neighborhood of 11×11 and subsampling (c3): *sum of Euclidean distances* (red) and *vote assignation* (green). No weighting is performed. The percentage obtained by *Correlation* is shown in blue.

If weighting is applied to the *independent reparametrization* procedure, the results can still improve a little bit: Figure 4.19 shows the degree of accuracy obtained by weighting the distances with the standard deviation computed from the interest point and all the candidate matches.

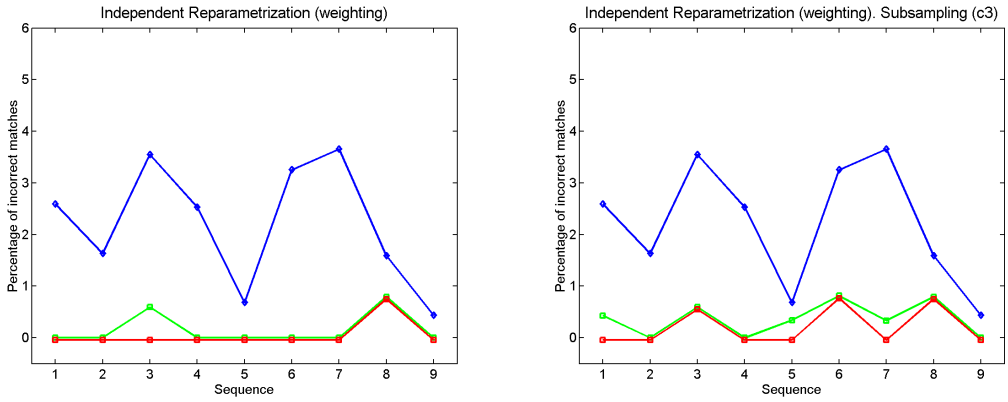


Figure 4.19. Percentage of incorrect matches (*independent reparametrization*), considering a neighborhood of 11×11 and subsampling (c3): *sum of Euclidean distances* (red) and *vote assignation* (green). Weighting is performed considering the interest point and all the candidate matches. *Correlation* is shown in blue.

4.4 Conclusions

The performance of different texture operators has been tested in underwater images. Due to the unstructured nature of the underwater environment and natural images in general, only non-structural texture operators have been considered. The results have proved that some configurations of the *Energy Filters* proposed by Laws are very adequate to describe underwater images. The performance of the *Contrast* operator has also proved to be very useful in this context.

As we expected, better results are obtained by characterizing the points with an 11×11 neighborhood, instead of a smaller one. Higher neighborhoods have been avoided since the hypothesis of negligible deformations of the pattern inside the window fails as the characterization window increases its size. As it was initially anticipated, the similarity measures which keep the information regarding the spatial distribution of the elements (like *Euclidean distance*) of the characterization vector perform better than the ones that fuse all the information in a single value (*mean* or *standard deviation*).

The experimental validation for selecting a strategy to fuse the information provided by every texture operator has shown that it is necessary a standarization process, since the different texture operators work different dynamic ranges. However, most of the methods which have been tested have provided satisfactory results (better than the classical correlation strategy). Concretely, we have selected

the following as the one which will be used to solve the problem of correspondence in our mosaicking algorithm:

> *Independent reparametrization of every characterization vector by taking into account the minimum and maximum value of the vector (Independent reparametrization), and considering a neighborhood of 11 × 11 pixels. Weighting is performed considering the interest point and all the candidate matches (shown in Figure 4.19).*

Subsampling with strategy (c3) is strongly recommended, since it is computationally efficient (only considers 9 points in the neighborhood), and the results are nearly as good as taking the whole set of neighbors to characterize the point.

Therefore, the result of applying the selected feature characterization to every candidate match is compared with the characterization vector of the interest point. Then, the Euclidean distance as defined by the selected reparametrization strategy is computed. Finally, the candidate correspondence with the higher similitude measurement (lower distance) is selected.

References

[Cuf98] X. Cufí, “Aportació a la segmentació d’imatges mitjançant l’obtenció dels contorns envolupants” (*in catalan*), Ph.D. Thesis, Politechnical University of Catalonia, Barcelona, Spain, 1998.

[Fau93] O. Faugeras, “Three-dimensional Computer Vision: A Geometric Viewpoint,” The MIT Press, Cambridge, Massachusetts, 1993.

[Fre00] J. Freixenet, “Descripció d’escenes exteriors a partir d’un aprenentatge supervisat de les característiques més significatives dels objectes” (*in catalan*), Ph.D. Thesis, University of Girona, Girona, Spain, 2000.

[Fun72] C.J. Funk, S.B. Bryant and P.J. Beckman Jr., “Handbook of underwater imaging system design”, Ocean Technology Department, Naval Undersea Center, 1972.

[Gar00] R. Garcia, X. Cufí, Ll. Pacheco, “Image Mosaicking for Estimating the Motion of an Underwater Vehicle”, *5th IFAC Conference on Manoeuvring and Control of Marine Crafts*, Aalborg, Denmark, 2000.

[Gar01a] R. Garcia, X. Cufí and J. Batlle, "Detection of Matchings in a Sequence of Underwater Images through Texture Analysis", *IEEE International Conference on Image Processing*, Thessaloniki, Greece, 2001.

[Gar01b] R. Garcia, X. Cufí, X. Muñoz, L. Pacheco, J. Batlle, "An Image Mosaicking Method based on the Integration of Grey Level and Textural Features", *IX Simposium Nacional de Reconocimiento de Formas y Análisis de Imágenes*, Benicassim, Castellón, 2001.

[Gar01c] R. Garcia, J. Batlle, X. Cufí and J. Amat, "Positioning An Underwater Vehicle Through Image Mosaicking," *IEEE International Conference on Robotics and Automation*, Seoul, Rep. of Korea, 2001.

[Gia00] A. Giachetti, "Matching techniques to compute image motion," *Image and Vision Computing*, no. 18, pp. 247–260, 2000.

[Har73] R.M. Haralick, K. Shanguman, and I. Dinstein, "Textural Features for image classification," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 3, pp. 610-621, November, 1973.

[Har95] D. Harwood, T. Ojala, M. Pietikäinen, S. Kelman and L. Davis, "Texture Classification by center-symmetric auto-correlation, using Kullback discrimination of distributions," *Pattern Recognition Letters*, vol. 16, pp. 1–10, 1995.

[Kle98] R. Klette, K. Schlüns and A. Koschan, "Computer Vision: three-dimensional data from images," Springer-Verlag, 1998.

[Law80] K.I. Laws, "Textured Image Segmentation," Ph.D. Thesis, Processing Institute, University of Southern California, Los Angeles, 1980.

[Mar98] J. Martí, "Aportació a la descripció d'escenes urbanes mitjançant aproximació de models" (*in catalan*), Ph.D. Thesis, Politechnical University of Catalonia, Barcelona, Spain, 1998.

[Oja96] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative Study of Texture Measures with Classification Based on Feature Distribution," *Pattern Recognition*, vol. 29, pp. 51–59, 1996.

[Oja99] T. Ojala and M. Pietikäinen, "Unsupervised texture segmentation using feature distributions," *Pattern Recognition*, vol. 32, pp. 477–486, 1999.

[Zha94] Z. Zhang, R Deriche, O. Faugeras and Q.T. Luong, "A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry," INRIA RR-2273, 1994.

Chapter 5

Proposal of a method to construct visual mosaics

In this chapter we present the new methodology to construct visual mosaics of the underwater environment. The proposed system first performs a correction of the geometric distortions produced by the camera lenses and the medium, and then takes profit of the texture-based correspondence detector presented in the previous Chapter to obtain a set of matched features. Next, a robust regression technique is used to estimate the planar transformation which explains the motion of the majority of the features. Then, this planar transformation is used to relate every image to the common mosaic frame, constructing a composite image with all the frames of the sequence. Finally, the procedure to obtain the 3D motion of the vehicle in the world reference frame is described.

5.1 Overview

Since the texture characterization described in the previous chapter has proved to obtain quite an acceptable ratio of correct correspondences in underwater images, a

feature-based mosaicking system can be constructed. The creation of the mosaic will be accomplished in the following stages (see Figure 5.1): First, a correction of the geometric distortion caused by the lens and the interface of the camera housing is performed. A detector of interest points then selects the most reliable features of the undistorted image and the correspondences of these features are matched in the next image of the sequence by means of the strategy proposed in Chapter 4. Next, the system identifies the points which describe the dominant motion of the image by means of a robust outlier-detection algorithm. Once the pairs of features describing the dominant motion have been selected, a 2D projective transformation matrix relating the coordinates of both images is computed. Finally, the registered images are merged onto a composite mosaic image, and absolute 3D position estimates can be obtained.

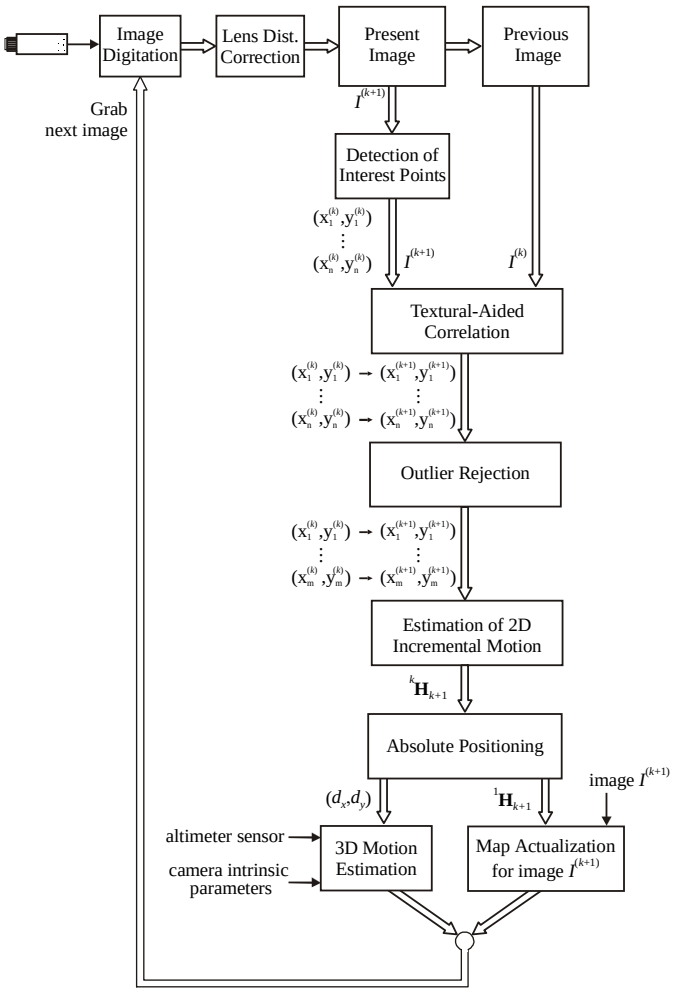


Figure 5.1. Block diagram of the proposed mosaicking system.

5.2 Lens distortion removal

Correcting the distortion produced by the camera lenses and the ray diffraction at the water-camera housing and the air-camera housing interfaces requires the estimation of a number of intrinsic camera parameters [Xu97]. A simplification of the Faugeras-Toscani algorithm has been implemented to correct uniquely radial distortion, instead of performing full camera calibration [Fau86]:

$$x_u = \left(\frac{x_d - x_0}{k_x} \right) + \left(\frac{x_d - x_0}{k_x} \right) \cdot k_1 \cdot r^2 + c_x \tag{5.1}$$

$$y_u = \left(\frac{y_d - y_0}{k_y} \right) + \left(\frac{y_d - y_0}{k_y} \right) \cdot k_1 \cdot r^2 + c_y \tag{5.2}$$

where (x_u, y_u) are the ideal undistorted coordinates of the measured distorted point (x_d, y_d), and (c_x, c_y) are the coordinates of the center of the image. The parameters k_x, k_y are the scaling factors in the *x* and *y* directions, respectively. They account for differences on the image axes scaling. The principal point of the image is defined by (x_0, y_0) and represents the coordinates of the projection of the optical center of the camera on the image plane. k_1 is the first term of the radial correction series, and r is the squared distance of (x_d, y_d) from the center of the image and accomplishes:

$$r = \sqrt{\left(\frac{x_d - x_0}{k_x} \right)^2 + \left(\frac{y_d - y_0}{k_y} \right)^2} \tag{5.3}$$

Once these parameters are known, image correction for radial distortion can be computed [Gar01d]. In order to compute the image correction parameters a planar calibration grid with a set of equally spaced black dots can be used. When the calibration grid is captured by the camera we obtain the coordinates of a set of *i* distorted points (x_d^i, y_d^i). By applying equations (5.1) and (5.2), we can obtain their undistorted pairs (x_u^i, y_u^i), and they can be compared with the target distortion-free coordinates of the image grid (x_t^i, y_t^i). Therefore, it is necessary to minimize the cost function $f(\cdot)$:

$$f\left(x_0, y_0, k_1, k_x, k_y\right) = \sum_i \sqrt{\left(x_u^i - x_t^i\right)^2 + \left(y_u^i - y_t^i\right)^2} \tag{5.4}$$

where the distance from the computed undistorted points to the target is to be minimized by adjusting the 5-parameter set $\left(x_0, y_0, k_1, k_x, k_y\right)$.

In our implementation, the undistorted values are computed offline. Then, two Look Up Tables (LUT) are stored with these values. At run time, the LUTs are addressed with the undistorted *x* and *y* coordinates and every LUT provides the distorted coordinate from which the gray-level has to be taken.

5.3 Selection of interest points

The goal of our interest point detector is to find stable features in the image, *i.e.* scene features which can reliably be found when the camera moves from one location to another and lighting conditions of the scene change somewhat. The strategy that has been used to detect these interest points is based on the corner detector proposed by Harris and Stephens [Har88]. This corner detector finds areas of high variation of the image gradient by using first-order image derivative approximations. It is based on the computation of the following matrix:

$$\mathbf{G} = \begin{bmatrix} \left\langle I_x^2 \right\rangle & \left\langle I_x I_y \right\rangle \\ \left\langle I_x I_y \right\rangle & \left\langle I_y^2 \right\rangle \end{bmatrix}, \text{ with } I_x = \left(\frac{\partial I}{\partial x}\right) \text{ and } I_y = \left(\frac{\partial I}{\partial y}\right) \tag{5.5}$$

where $\left\langle I_x I_y \right\rangle$ is the point-to-point product of images I_x and I_y, and $\langle\ \rangle$ indicates 2D Gaussian smoothing. The matrix **G** encodes the so-called local autocorrelation function. According to [Har88] this Gaussian filtering should be performed before edge extraction. It should be noted that this approach is quite similar to the Shi-Tomasi tracker [Shi94]. A feature is a good candidate to track if **G** is well-conditioned, that is, if both eigenvalues of this matrix are large. This means that the image point in which the partial derivatives have been computed presents a rapid intensity variation on neighboring pixels in the *x* and *y* directions. Therefore, this point can be considered a corner. The corner response function *R* is given in equation (5.6). This quantity measures the rate of change of gradient at every point of the image and is regarded as a “cornerness” measure.

$$R = \frac{k}{(k+1)^2}\left(\det(\mathbf{G}) - \left(\text{trace}(\mathbf{G})\right)^2\right) \tag{5.6a}$$

$$R = \frac{k}{(k+1)^2}\left(\left[\left\langle I_x^2\right\rangle\left\langle I_y^2\right\rangle - \left\langle I_x I_y\right\rangle^2\right] - \left[\left\langle I_x^2\right\rangle + \left\langle I_y^2\right\rangle\right]^2\right) \tag{5.6b}$$

In [Har88] the Gaussian smoothing is proposed to be a circular window. The value *k* in the response function is the maximum ratio of eigenvalues of **G** for which the response function is positive. Harris and Stephens [Har88] suggest a value of 25 to be used. In our implementation we consider a 3×3 Gaussian mask for smoothing. However, this mask is applied after the computation of the partial derivatives, as can be seen in Figure 5.2. On the other hand, a simplified version of the "cornerness" measure is considered, avoiding the selection of parameter *k*. Equation (5.7) shows the alternative computation of the "cornerness" measure (*C*).

$$C = \frac{I_x^2 + I_y^2}{I_x^2 I_y^2 - I_{xy}^2} \tag{5.7}$$

The "cornerness" is computed for all the pixels of the image. High values of parameter *C* imply poor image gradient. On the contrary, as it gets smaller the "cornerness" of this area augments. Then, for a considered neighborhood, any values which are not local minimum are suppressed. This leads to a sparse corner point map for each image. Finally, since *C* provides a measure of the quality of the corner, a descending sorting is performed. In this way, the algorithm is able to provide the best *n* interest points of the image.

```
ALGORITHM Interest_Point_Detector
    Ix  = CONVOLVE (Image, Prewitt_Hor / 3)
    Iy  = CONVOLVE (Image, Prewitt_Ver / 3)
    Ixy = Ix * Iy
    Ix_Sq = Ix^2
    Iy_Sq = Iy^2
    Ix  = CONVOLVE ( Ix_Sq, Gaussian )
    Iy  = CONVOLVE ( Iy_Sq, Gaussian )
    Ixy = CONVOLVE ( Ixy, Gaussian )
    C = (Ix + Iy) / ((Ix * Iy) – Ixy^2)
    FOR Y = 1 TO ImsizeY DO
      FOR X = 1 TO ImsizeX DO
           Define Local Minimum Neighborhood ( C )
           Min(Y,X) = Minimum Neighbor ( C )
      ENDFOR
    ENDFOR
    FOR Y = 1 TO ImsizeY DO
      FOR X = 1 TO ImsizeX DO
           IF Min(Y,X) = C(Y,X) THEN
                 Store Point (Y,X, C(Y,X))
           ENDIF
      ENDFOR
    ENDFOR
    Descending Sort according to C
    Return Best n Points
ENDALGORITHM
```

Figure 5.2. Interest Point Detector algorithm

5.4 Detection of correspondences

Once the interest points have been detected in the one of the images, the next step consists in finding their correspondences in the next one. Therefore, we propose the use of the strategy discussed in Chapter 4, which can be summarized as follows. Given an interest point **m** in image I, a $m \times m$ region centered at this point is selected as correlation window. This window is subsampled as indicated in equation (4.1) (page 90) according to the parameter q[1]. Then, a *search window* is defined in image I'. This window is normally centered at the coordinates of the interest point. Then, the correlation score (CS) is computed for all the points of the search window. The set of matches with a similarity higher than 85% $(\mathrm{CS} \geq 0.7)$ are added to the list of candidate correspondences of **m**. The interest point and its L possible

[1] Our default parameters are m=21 and q=2.

correspondences are then characterized by means of a characterization vector (CV). Nine texture operators will form every vector [Gar01a]. Every texture operator has nine measures, for the nine neighbors of the point which is being characterized, considering a neighborhood of 11 × 11 pixels, and subsampling strategy (c3) of Figure 4.11. The characterization vectors are then reparametrized independently, taking into account the minimum and maximum value of the vector in question. Then, the Euclidean distance is computed between the CV of the interest point and every candidate match, as defined in equation (4.7) (page 93). Finally, every distance is weighted considering the interest point and all the candidate matches, as indicated in equation (4.21) (page 115). Finally, the candidate correspondence with the smaller distance is selected as the correct match $\mathbf{m}'_i$.

Further details of the above algorithm can be found in Chapter 4. This operation has to be performed for all the candidate matches. Unfortunately, computing the CS for all the points of the search window can be quite time consuming. For this reason a series of optimizations are proposed below.

The size of the search window depends on the cycle time of the algorithm and the apparent velocity at which the camera moves. Since the camera is mounted on a UUV, we can assume that the motion of the camera is undergoes smooth changes of velocity and direction. In this case, the search window can be centered at the position predicted by the transformation matrix **H** which describes the motion of the vehicle in the previous time instant. Therefore, the size of the search window can be reduced, thus saving computing time.

A second optimization can be obtained if not all the pixels of the search window in I' are correlated with the correlation window of *I*. We know that the matched point is the projection of an interest point, which is a point with a high gradient contrast in both directions of the image axis. Then, after the interest point detection has been performed in image *I*, a subsequent detection of interest points has to be performed in the second image I'. This second detection of interest points should be performed with a lower threshold of the “cornerness” parameter, therefore detecting more interest points than in the first image. And the values which are not local minimum should not be suppressed, thus ensuring that the interest points of the first image have been selected. Next, the search window is defined in I', but now only those points which have been selected as interest points are considered for

correlation. Finally, the list of candidate matches will be formed by the interest points of the second image whose CS is above the threshold, as described before.

5.5 Outlier removal and homography estimation

5.5.1 Introduction

After the correspondences have been solved, a set of displacement vectors relating the features of two images of the sequence is obtained. Every vector relates the coordinates of the same feature in both images. Our aim is now to recover the apparent motion of the camera from these features. This can be done by computing a 2D transformation matrix **H** which relates the coordinates of a feature in a frame with its coordinates in the previous one:

$$\tilde{\mathbf{m}} = \mathbf{H} \cdot \tilde{\mathbf{m}}' \text{ or } \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \tag{5.8}$$

where $\tilde{\mathbf{m}} = (x_i, y_i, 1)^T$ and $\tilde{\mathbf{m}}' = (x_i', y_i', 1)^T$ denote a correspondence point in two consecutive images; the symbol ~ indicates that the points are expressed in homogeneous coordinates, and $\cong$ expresses equality up to scale. The matrix **H** that performs this transformation is known as “homography” [Sem52]. Since this matrix is defined up to scale, equation (5.8) can be written as follows:

$$\begin{bmatrix} kx_i \\ ky_i \\ k \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \tag{5.9}$$

where k is an arbitrary non-zero constant. By using inhomogeneous coordinates instead of the homogeneous coordinates of the points, the projective transformation of equation (5.9) can be written as:

$$\left.\begin{aligned} x_i &= \frac{h_{11}x'_i + h_{12}y'_i + h_{13}}{h_{31}x'_i + h_{32}y'_i + 1} \\ y_i &= \frac{h_{21}x'_i + h_{22}y'_i + h_{23}}{h_{31}x'_i + h_{32}y'_i + 1} \end{aligned}\right\} \tag{5.10}$$

Each point correspondence generates two equations for the elements of **H**, which after multiplying out are:

$$\left.\begin{aligned} \left(h_{31}x'_i + h_{32}y'_i + 1\right)x_i &= h_{11}x'_i + h_{12}y'_i + h_{13} \\ \left(h_{31}x'_i + h_{32}y'_i + 1\right)y_i &= h_{21}x'_i + h_{22}y'_i + h_{23} \end{aligned}\right\} \tag{5.11}$$

Operating the terms, the following linear system in the terms of **H** can be obtained:

$$\left.\begin{aligned} x_i &= h_{11}x'_i + h_{12}y'_i + h_{13} - h_{31}x_i x'_i - h_{32}x_i y'_i \\ y_i &= h_{21}x'_i + h_{22}y'_i + h_{23} - h_{31}y_i x'_i - h_{32}y_i y'_i \end{aligned}\right\} \tag{5.12}$$

which expressed in matricial form and considering *n* pairs point/matching gives rise to:

$$\begin{bmatrix} x'_1 & y'_1 & 1 & 0 & 0 & 0 & -x_1 \cdot x'_1 & -x_1 \cdot y'_1 \\ 0 & 0 & 0 & x'_1 & y'_1 & 1 & -y_1 \cdot x'_1 & -y_1 \cdot y'_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_n & y'_n & 1 & 0 & 0 & 0 & -x_n \cdot x'_n & -x_n \cdot y'_n \\ 0 & 0 & 1 & x'_n & y'_n & 1 & -y_n \cdot x'_n & -y_n \cdot y'_n \end{bmatrix} \cdot \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ \vdots \\ x_n \\ y_n \end{bmatrix} \tag{5.13}$$

or $\mathbf{A} \cdot \mathbf{h} = \mathbf{b}$. Therefore, the homography matrix **H** can be computed from equation (5.13) if 4 or more pairs of matchings are available. The 8 unknowns in **h** could be solved by means the pseudo-inverse least squares strategy:

$$\mathbf{h} = \left(\mathbf{A}^{\mathrm{T}} \cdot \mathbf{A}\right)^{-1} \mathbf{A}^{\mathrm{T}} \cdot \mathbf{b} \tag{5.14}$$

However, this involves the inversion of the matrix $\left(\mathbf{A}^{\mathrm{T}} \cdot \mathbf{A}\right)^{-1}$. Unfortunately, this operation could fail to give satisfactory results, since this matrix could be either singular of very close to singular. For this reason we use the technique known as *Singular Value Decomposition* (SVD). SVD methods state that any *m* × *n* matrix **A** whose number of rows *m* is greater than of equal to its number of columns *n*, can be written as the product of an *m* × *n* column-orthogonal matrix **U**, an *n* × *n* diagonal

matrix **W** with positive or zero elements (the *singular values*), and the transpose of an $n \times n$ orthogonal matrix **V** [Gol89].

$$\mathbf{A}_{m\times n} = \mathbf{U}_{m\times n} \cdot \mathbf{W}_{n\times n} \cdot \mathbf{V}^{\mathrm{T}}_{n\times n} \tag{5.15}$$

Then, the SVD solution of equation (5.13) can be obtained from:

$$\mathbf{h} = \sum_{i=1}^{m} \left(\frac{\mathbf{U}_{(i)} \cdot \mathbf{b}}{\mathbf{W}_{(i,i)}} \right) \mathbf{V}_{(i)} \tag{5.16}$$

where $\mathbf{U}_{(i)}$ $i = 1,...,n$ denote the columns of **U** (each one a vector of *m* elements); $\mathbf{V}_{(i)}$ $i = 1,...,n$ are the columns of **V** (each one a vector of length *n*); and $\mathbf{W}_{(i,i)}$ $i = 1,...,n$ denote the singular values of matrix **A**, as calculated in equation (5.15).

5.5.2 Data normalization

The accuracy of the estimated homography **H** also depends on the coordinate frame in which the points are expressed. We have applied a method of normalization based on [Har97]. The normalization procedure is independently performed for the interest points $\{\tilde{\mathbf{m}}_1, \tilde{\mathbf{m}}_2, ..., \tilde{\mathbf{m}}_n\}$ in image *I* and the matches $\{\tilde{\mathbf{m}}'_1, \tilde{\mathbf{m}}'_2, ..., \tilde{\mathbf{m}}'_n\}$ in image I'.

The normalization for every image is accomplished as follows. First, the coordinates in each image are translated (by a different translation of each image), bringing the centroid $(\bar{x}, \bar{y})$ of the set of points $\{\tilde{\mathbf{m}}_1, \tilde{\mathbf{m}}_2, ..., \tilde{\mathbf{m}}_n\}$ to the origin of coordinates. Considering $\tilde{\mathbf{m}}_i = (x_i, y_i, 1)^T$), the centroid can be computed from:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \quad \text{and} \quad \bar{y} = \frac{\sum_{i=1}^{n} y_i}{n} \tag{5.17}$$

Then, the coordinates have to be scaled so that the average distance from a point to the origin is $\sqrt{2}$. To perform this change of scale, we should know the initial average distance $(\bar{d})$ from every point to the origin of coordinates:

$$\bar{d} = \frac{\sum_{i=1}^{n} \sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2}}{n} \tag{5.18}$$

and then the scaling factor *s* can be computed from $\bar{d} \cdot s = \sqrt{2}$. The translation and scaling can be performed by means of the transformation matrix **T**.

$$\mathbf{T} = \begin{pmatrix} \frac{\sqrt{2}}{\bar{d}} & 0 & -\left(\frac{\sqrt{2}}{\bar{d}}\bar{x}\right) \\ 0 & \frac{\sqrt{2}}{\bar{d}} & -\left(\frac{\sqrt{2}}{\bar{d}}\bar{y}\right) \\ 0 & 0 & 1 \end{pmatrix} \tag{5.19}$$

Since the same type of normalization is applied to the matches $\{\tilde{\mathbf{m}}'_1, \tilde{\mathbf{m}}'_2, ..., \tilde{\mathbf{m}}'_n\}$ a transformation $\mathbf{T}'$ can be computed:

$$\mathbf{T}' = \begin{pmatrix} \frac{\sqrt{2}}{\bar{d}'} & 0 & -\left(\frac{\sqrt{2}}{\bar{d}'}\bar{x}'\right) \\ 0 & \frac{\sqrt{2}}{\bar{d}'} & -\left(\frac{\sqrt{2}}{\bar{d}'}\bar{y}'\right) \\ 0 & 0 & 1 \end{pmatrix} \tag{5.20}$$

Therefore, given an interest point $\tilde{\mathbf{m}}$ and its correspondence $\tilde{\mathbf{m}}'$, the normalized coordinates $\tilde{\mathbf{m}}'_N$ and $\tilde{\mathbf{m}}'_N$ can be obtained by applying transformations $\mathbf{T}$ and $\mathbf{T}'$:

$$\tilde{\mathbf{m}}_N = \mathbf{T}\tilde{\mathbf{m}} \tag{5.21}$$

$$\tilde{\mathbf{m}}'_N = \mathbf{T}'\tilde{\mathbf{m}}' \tag{5.22}$$

Then, equation (5.13) can be applied to find an homograhy $\mathbf{H}_N$ from the normalized data. Finally, a denormalization is required to obtain the "standard" homography $\mathbf{H}$:

$$\mathbf{H} = (\mathbf{T})^{-1}\mathbf{H}_N\mathbf{T}' \tag{5.23}$$

In summary, the estimation of the homography matrix $\mathbf{H}$ a set of points $\{\tilde{\mathbf{m}}_1, \tilde{\mathbf{m}}_2, ..., \tilde{\mathbf{m}}_n\}$ and correspondences $\{\tilde{\mathbf{m}}'_1, \tilde{\mathbf{m}}'_2, ..., \tilde{\mathbf{m}}'_n\}$ is detailed in Figure 5.3:

5.5.3 Detection of outliers

Although an accurate texture analysis is devoted to the matching procedure, some false matches (known as *outliers*) could still appear among the right correspondences. For this reason, a robust estimation method has to be applied. The *Least Median of Squares* (LMedS) algorithm can be used for finding the matrix $\mathbf{H}$ which minimizes the median of the squared residuals M_{err} [Gar01b]:

$$M_{err} = \underset{j}{med}\left(d^2\left(\tilde{\mathbf{m}}_j, \mathbf{H}\tilde{\mathbf{m}}'_j\right) + d^2\left(\tilde{\mathbf{m}}'_j, \mathbf{H}^{-1}\tilde{\mathbf{m}}_j\right)\right) \qquad \forall j \tag{5.24}$$

where $\tilde{\mathbf{m}} = (\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3)$ are the homogeneous coordinates of a 2D point $\mathbf{m}$ defined in the image plane *I*, being $\mathbf{m} = (x_i, y_i) = (\mathrm{x}_1/\mathrm{x}_3, \mathrm{x}_2/\mathrm{x}_3)$ its corresponding Cartesian

coordinates; and $d^2\left(\tilde{\mathbf{m}}_j, \mathbf{H}\tilde{\mathbf{m}}'_j\right)$ is the square distance from a point $\tilde{\mathbf{m}}_j$, defined on image *I*, to the projection on the same image plane of its correspondence $\tilde{\mathbf{m}}'_j$. Hence, the error residual M_{err} is defined by the distance of a point to the projection of its correspondence [Rou87], and vice versa, the distance from the correspondence to the point, considering the inverse homography $\mathbf{H}^{-1}$.

ALGORITHM Hom_Estimation_from_Normalized_Data

1. Compute the centroid of the set of points $(\overline{x}, \overline{y})$ and the set of matches $(\overline{x}', \overline{y}')$, as shown in equation (5.17).
2. Compute the average distance from every point to the origin of coordinates, as shown in equation (5.18). A distance is obtained for the interest points (d), and the correspondences (d').
3. Compute the matrices $\mathbf{T}$ and $\mathbf{T}'$, as in equations (5.19) and (5.40).
4. Find a set of normalized points $\{\tilde{\mathbf{m}}_{1N}, \tilde{\mathbf{m}}_{2N}, ..., \tilde{\mathbf{m}}_{nN}\}$ and correspondences $\{\tilde{\mathbf{m}}'_{1N}, \tilde{\mathbf{m}}'_{2N}, ..., \tilde{\mathbf{m}}'_{nN}\}$ by applying transformations $\mathbf{T}$ and $\mathbf{T}'$, as shown in equations (5.21) and (5.22).
5. Compute a homography matrix $\mathbf{H}_N$ from the normalized points, though SVD.
6. Compute the homography $\mathbf{H}$ from $\mathbf{H}_N$, $\mathbf{T}$ and $\mathbf{T}'$, as shown in equation (5.23).

ENDALGORITHM

Figure 5.3. Algorithm to compute the homography $\mathbf{H}$ normalizing the image coordinates.

The LMedS algorithm works as follows: given the regression problem of computing the matrix **H** from a set of data points, compute a candidate solution based on a randomly chosen *d*-tuple from the data. Then, estimate the fit of this solution to all the data, defined as the median of the squared residuals M_{err}. The amount of randomly chosen data *d* may vary depending on the motion model which is used in the mosaic. For instance, in the case of a general projective homography where 8 parameters have to be estimated *d* has a value of 4, while in the case of an affine homography only 6 parameters have to be found, reducing *d* to 3.

The LMedS algorithm has a *breakdown point* of 50%, *i.e.*, the algorithm can tolerate up to 50% of outliers, but not more. In principle, all the *d*-tuples should be evaluated; in practice a Monte Carlo technique is applied, in which only a random sample of size *r* is considered. Assuming that the whole set of points may contain up to a fraction of ε outliers, the probability that at least one of the *k d*-tuples consists of *d* "inliers" (correct data) is given by:

$$P = 1 - \left(1 - (1-\varepsilon)^d\right)^k \tag{5.25}$$

Therefore, *k* can be found from:

$$k = \frac{\log(1-P)}{\log\left(1-(1-\varepsilon)^d\right)} \tag{5.26}$$

Table 5.1 shows the number of samples required to ensure, with a probability *P*=0.99, that at least one sample has no outliers for different sample sizes (*d*) and percentage of outliers (ε).

Table 5.1. Number *k* of samples required to have a probability of 0.99 that at least one sample has no outliers.

Sample Size	*Proportion of outliers* (ε)					
d	5%	10%	20%	30%	40%	50%
2	2	3	5	7	11	17
3	3	4	7	11	19	35
4	3	5	9	17	34	72

It can be seen from the table that considering the worst case, where 50% of the data are outliers, the affine model would require 35 samples, while the projective model goes up to 72.

As noted in [Rou87], the LMedS efficiency is poor in the presence of Gaussian noise, considering the *efficiency* of a method as the ratio between the lowest achievable variance for the estimated parameters and the actual variance provided by the given method. To compensate for this deficiency, we further carry out a weighted least-squares procedure. The *robust standard deviation* estimate $(\hat{\sigma})$ is obtained through:

$$\hat{\sigma} = 1.4826\left[1+\frac{5}{n-d}\right]\sqrt{M_{err}} \tag{5.27}$$

where M_{err} is the minimal median as defined in equation (5.24). Once the robust standard deviation $\hat{\sigma}$ is known, a weight can be assigned to each correspondence:

$$w_i = \begin{cases} 1 & \text{if } r_i^2 \le (2.5\hat{\sigma})^2 \\ 0 & \text{otherwise,} \end{cases} \tag{5.28}$$

where

$$r_i^2 = d^2\left(\tilde{\mathbf{m}}_i, \mathbf{H}\tilde{\mathbf{m}}_i'\right) + d^2\left(\tilde{\mathbf{m}}_i', \mathbf{H}^{-1}\tilde{\mathbf{m}}_i\right) \tag{5.29}$$

The pairs of interest point/correspondence having $w_i = 0$ are considered outliers. Therefore, in our implementation, those points at a distance larger than 2.5 times the robust standard deviation are eliminated, and matrix **H** is recomputed with the remaining points. In this was new homography matrix **H** is robustly estimated.

5.6 Image warping and mosaic construction

As soon as the best transformation **H** between two frames has been found, the two images could be warped together. According to our definition of matrix **H** (see equation (5.8), page 131), given the homogeneous coordinates of a point $\tilde{\mathbf{m}}'$ in image I', the product $\mathbf{H}\tilde{\mathbf{m}}'$ provides the coordinates of this point ($\tilde{\mathbf{m}}$) in image *I*. However, it is necessary to find a transformation which maps the present image with the mosaic image. The successive frame-to-frame transformations can be combined to form a global model, where all the frames are mapped into a common, arbitrarily chosen, reference frame, as shown in Figure 5.4.

Normally, the first image of the sequence is chosen as reference frame. Then, the set of incremental transformation matrices of the form ${}^{k}\mathbf{H}_{k+1}$ can be cascaded together to form the global model ${}^{1}\mathbf{H}_{k+1}$, which relates the $(k+1)^{th}$ image with the first image of the sequence.

$${}^{1}\mathbf{H}_{k+1} = \prod_{i=1..k} {}^{i}\mathbf{H}_{i+1} \tag{5.30}$$

Once the matrix ${}^{1}\mathbf{H}_{k+1}$ has been computed, image $I^{(k+1)}$ can be added to the mosaic image, since all the pixels of $I^{(k+1)}$ hold:

$$\tilde{\mathbf{m}}^{(1)} = {}^{1}\mathbf{H}_{k+1} \cdot \tilde{\mathbf{m}}^{(k+1)} \tag{5.31}$$

By measuring the relative position of image $I^{(k+1)}$ with respect to the mosaic reference frame a measure of the vehicle motion is obtained, at the same time that a visual map is constructed. Figure 5.4 illustrates the relationship between every consecutive homography and the mosaic frame [Gar00,Gar01c].

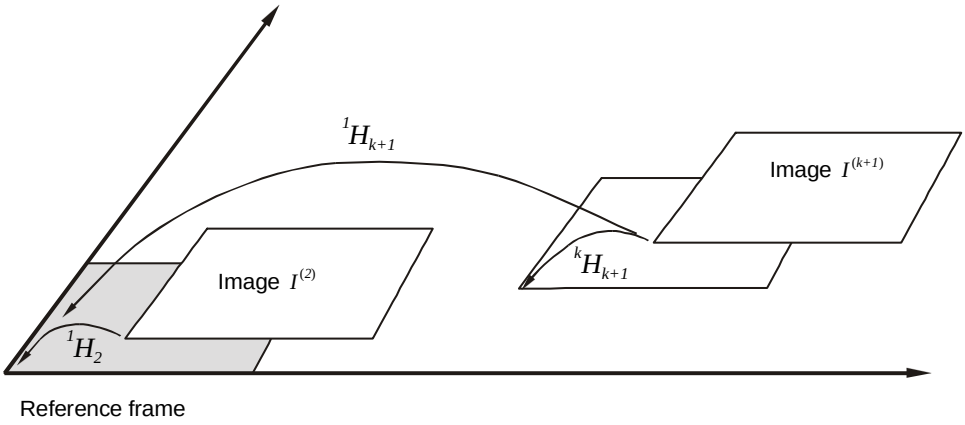


Figure 5.4. Mosaic common reference frame. The global registration matrix $^1\mathbf{H}_{k+1}$ relates the image coordinates of any point in image $I^{(k+1)}$ with respect of the coordinate frame of the first image $I^{(1)}$.

5.7 Motion estimation

The aim of this phase is to estimate the position of the vehicle as the mosaic is being constructed, following the Concurrent Mapping and Localization (CML) paradigm. At this point, the 2D motion of the camera is known in pixels from one image to the next, as an affine or projective measure (rotation, translation, scaling, shear, etc.). With the aid of an ultrasonic altimeter, and the knowledge of the intrinsic parameters of the camera, 3D metric information about vehicle motion can be recovered. This is done in the following way. As the distortion produced by the camera lenses and the ray diffractions at the air/camera housing/water interfaces has been corrected in the first phase of the mosaicking process, the processed images are an ideal projective projection of the ocean floor. That is, the camera behaves as a perfect *pin-hole* model, producing an ideal linear projection of the incident image rays, as shown in Figure 5.5.

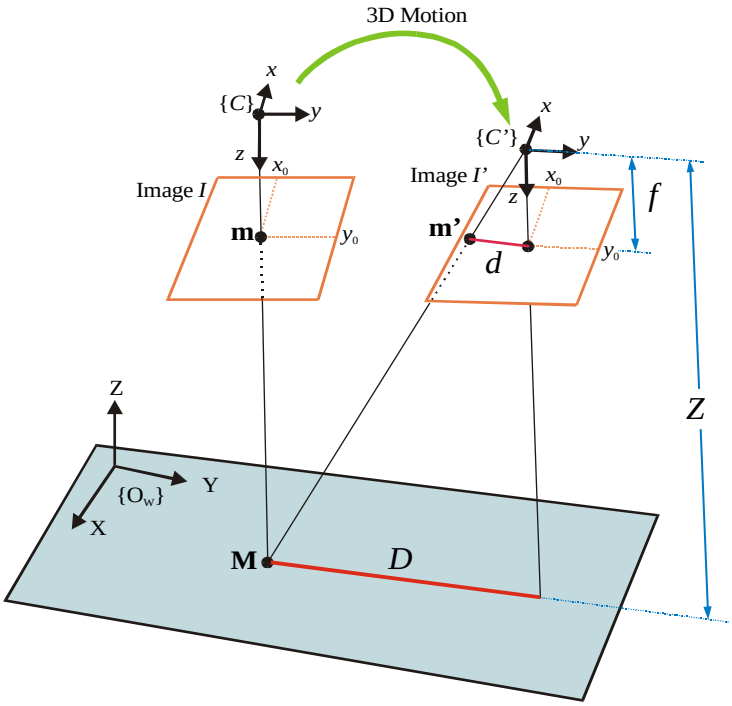


Figure 5.5. Motion estimation in world (metric) coordinates. The incremental motion *d* is obtained in pixels from the mosaic. Taking *Z* from the altimeter sensor, and knowing the camera focal length *f*, a measure *D* can be obtained in world coordinates.

Therefore, the metric measure *Z* provided by the altimeter, together with the knowledge of the camera focal length *f*, can be used to convert the incremental motion estimation from the camera coordinated system (in pixels) to the world reference system (metric information). Applying the geometric law of the perspective relation [Kan91], the following equation can be obtained:

$$\frac{d}{f} = \frac{D}{Z}, \text{ then } D = \frac{d \cdot Z}{f} \tag{5.32}$$

When the first image of the sequence is placed mosaic in the mosaic, the world coordinate system $\{O_w\}$ is aligned to the *XY* plane defined by this image, and the initial *Z* is measured from the altimeter. For every new image, the subsequent homographies provide a 2D estimation of the vehicle motion. Considering the picture illustrated in Figure 5.4, incremental measure *d* can be decomposed in d_x and d_y, measured with respect to the coordinate system of the previous image. Therefore, equation (5.32) can be decomposed in

$$D_x = \frac{d_x \cdot Z}{f}; \qquad D_y = \frac{d_y \cdot Z}{f} \tag{5.33}$$

where (D_x, D_y) are the components of the incremental motion from image I to I', expressed in world coordinates [Gar01d].

Therefore, the 3D position of the vehicle can be obtained from incremental motion (D_x, D_y) and absolute measure *Z*. This metric information relative to the sea bed can be very useful for navigation and mission planning.

References

[Fau86] O.D. Faugeras and G. Toscani, “The calibration problem for stereo”, in Proceedings *of the IEEE Computer Vision and Pattern Recognition*, pp. 15–20, 1986.

[Gar00] R. Garcia, X. Cufí, Ll. Pacheco, “Image Mosaicking for Estimating the Motion of an Underwater Vehicle”, *5th IFAC Conference on Manoeuvring and Control of Marine Crafts*, Aalborg, Denmark, 2000.

[Gar01] R. Garcia, J. Batlle, X. Cufi and J. Amat, “Positioning an Underwater Vehicle through Image Mosaicking,” in *Proceedings of the IEEE Int. Conf. on Robotics and Automation*, Seoul, Korea, vol. 3, pp. 2779–2784, 2001.

[Gar01a] R. Garcia, X. Cufí and J. Batlle, "Detection of Matchings in a Sequence of Underwater Images through Texture Analysis", *IEEE International Conference on Image Processing*, Thessaloniki, Greece, 2001.

[Gar01b] R. Garcia, X. Cufí, X. Muñoz, L. Pacheco, J. Batlle, “An Image Mosaicking Method based on the Integration of Grey Level and Textural Features”, *IX Simposium Nacional de Reconocimiento de Formas y Análisis de Imágenes*, Benicassim, Castellón, 2001.

[Gar01c] R. Garcia, J. Batlle, X. Cufí and J. Amat, “Positioning An Underwater Vehicle Through Image Mosaicking,” *IEEE International Conference on Robotics and Automation*, Seoul, Rep. of Korea, 2001.

[Gar01d] R. Garcia, X. Cufi and M. Carreras “Estimating the Motion of an Underwater Robot from a Monocular Image Sequence”, *IEEE Conference on Robots and Systems*, Hawaii, *to be published*.

[Gol89] G.H. Golub and C.F. van Loan, “Matrix Computations”, Johns Hopkins University Press, 2nd Ed., Baltimore, 1989.

[Gra00] N. Gracias and J. Santos-Victor, "Underwater Video Mosaics as Visual Navigation Maps," *Computer Vision and Image Understanding*, vol. 79, no. 1, pp. 66–91, 2000.

[Har88] C.G. Harris and M.J. Stephens, "A combined corner and edge detector," in *Proceedings of the Fourth Alvey Vision Conference*, Manchester, pp. 147–151, 1988.

[Har97] R.I. Harley, "In defense of the eight-point algorithm", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 6, pp. 580–593, 1997.

[Kan91] K. Kanatani, "Computational projective geometry", *International Journal of Computer Vision*, Graphics and Image Processing, vol. 54, no. 3, pp. 333-348, 1991.

[Rou87] P. Rousseeuw and A. Leroy, "Robust Regression and Outlier Detection", John Wiley & Sons, New York, 1987.

[Sem52] J.G. Semple and G.T. Kneebone, "Algebraic projective geometry," Oxford University Press, 1952.

[Shi94] J. Shi and C. Tomasi "Good features to track", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600, 1994.

[Xu97] X. Xu and S. Negahdaripour, "Vision-based motion sensing from underwater navigation and mosaicing of ocean floor images," in *Proceedings of the MTS/IEEE OCEANS*, vol.2, pp. 1412–1417, 1997.

Chapter 6

Mosaic correction from crossover paths

This Chapter describes a methodology that can be used to obtain trajectory estimates from information of the crossover trajectories in the mosaic. As the mosaic increases in size, image local alignment errors increase the inaccuracies associated to the position of the vehicle. When the arbitrary path of the submersible describes a loop, the images forming the mosaic are re-aligned and a better position estimation is obtained. Kalman filtering appears as an extraordinary framework to deal with position estimates and their associated covariance.

6.1 Introduction

In the previous section a feature-based mosaicking strategy has been proposed to estimate the position of an underwater vehicle, while a visual map is being constructed. When performing a mission, the submersible follows an arbitrary path. Then, the mosaicking system will construct a mosaic of the surveyed area, reconstructing the trajectory followed by the vehicle. As the mosaic increases in size, image local alignment errors increase the error margin associated to the position of the vehicle. Even though the relative motion measured between images is very

precise, the error in absolute position increases indefinitely because of the small errors that accumulate over the length of the vehicle path. Occasionally, this path may cross over itself. In this situation new information is available, and the system can readjust the position estimates. Chapter 3 already introduced the work of Fleischer *et al.* [Fle96,Fle97] which was also based on the idea of reducing the vehicle drift when it revisits an already mosaicked area. However, their approach suffers from some limitations, *e.g.*, those derived from the batch methods, such as the requirement of all the data (present and future) before optimization can start.

In order to describe our proposal, consider the situation of Figure 6.1, where the vehicle describes an arbitrary path. The mosaicking system estimates the motion of the submersible by registering every pair of consecutive images. At time step (*i*), let the variance of the vehicle position σ_i^2, *i.e.*, statistical inaccuracy about where exactly the vehicle is. When image (*i*+1) is added to the mosaic, a small misalignment error is introduced. Therefore, the associated variance at time step (*i*+1) becomes σ_{i+1}^2, with $\sigma_{i+1}^2 > \sigma_i^2$. As vehicle moves, the variance of its position augments. Therefore, at time step (*k*), after registering images $(k-1)$ and (*k*), the variance σ_k^2 can be reduced if this image is directly registered to the mosaic. In this way, an inconsistency can be detected in the mosaic: the center point of image (*k*), denoted **p** in the Figure, corresponds to the same scene point which was located in image (*i*), and denoted $\mathbf{p}'$. Therefore, vehicle position can be corrected and the uncertainty on the location of the vehicle can be reduced to the existing uncertainty when image (*i*) was added to the mosaic. However, it is not sufficient to simply update the current vehicle position, reduce the error variance $(\sigma_k^2 = \sigma_i^2)$ and continue the construction of the mosaic. It is necessary to propagate this improvement back through the image chain, to improve the placement of all images within the mosaic. In this manner, two objectives are achieved:

1. The variance associated to the position of the images within the loop is reduced. Therefore, if a new crossover of the vehicle path is detected later in the area between image (*i*) and image (*k*), a lower variance would be obtained and, therefore, a better estimation of the vehicle position.

2. The visual quality of the mosaic will be improved, solving the inconsistencies in the mosaic. Therefore, the final map would be more adequate for positioning in future missions.

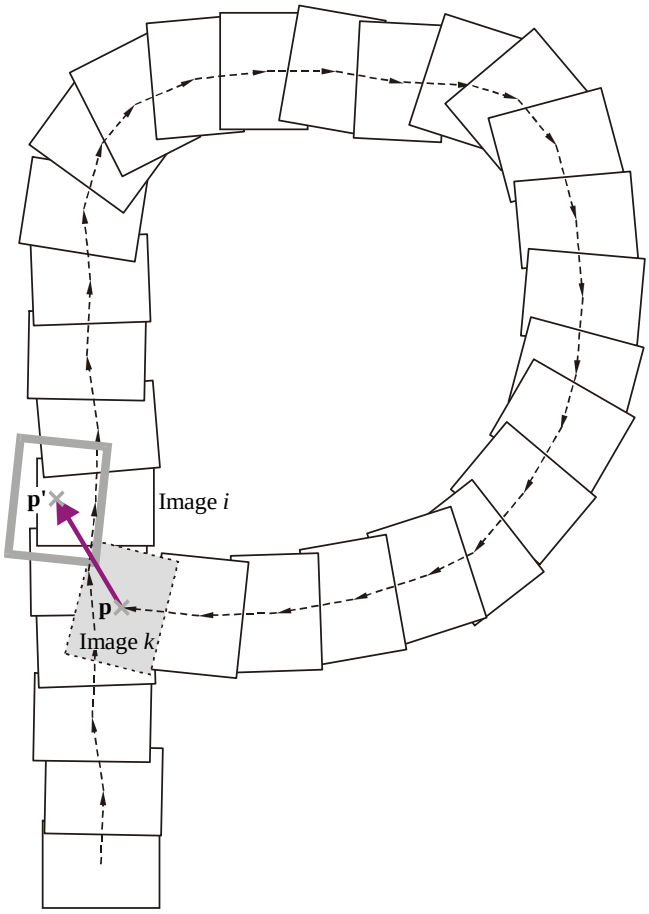


Figure 6.1: Arbitrary mosaic describing a crossing path.

Fleischer *et al.* proposed in [Fle96, Fle97] a continuous optimal estimation theory, so as to reduce the location error whenever the vehicle path crosses itself. In [Fle96] the smoother filter was applied in a discrete fashion, assuming that the local displacements were constant between consecutive images. Unfortunately, this assumption was difficult to achieve in practice, as the acquisition of a new image before the vehicle has moved the desired displacement gave the system a higher degree of robustness. Moreover, the derivation and implementation of the discrete algorithm when multiple loops of the vehicle are present is more difficult than its

derivation in the continuous scenario. For this reason, the same authors later proposed a continuous version of its smoother filter [Fle97], preventing the system from experiencing the problems described above.

The smoother filter implemented by MBARI/Stanford researchers assumes that the errors accumulate smoothly all over the loop. However, in practical situations the errors in building the mosaic are not distributed uniformly across the mosaic. On the contrary, at some points the error can be much larger than at other points, even though the line where the images are joined together at their edges has good visual registration. We solve this problem by introducing a measurement of the estimated variance of the image placement.

6.2 Crossover detection

It has been seen that drift can be corrected when the vehicle revisits a previously mapped zone, taking advantage of the extra positional information gained with the loop. Therefore, one of the first tasks that have to be achieved is the detection of crossover points in the image sequence. The straight solution is to take into account how the error variance propagates as the mosaic increases in size, as shown in Figure 6.2. This Figure gives an intuitive evolution of the error variance of the current image through the *error variance window*. This window represents the bounded area in which the image is located. To determine if a crossover has occurred, the system checks whether the area covered by the error variance of the current image intersects the mosaic image in an already surveyed zone. Indeed, crossover detection can only be accomplished if the present image and the overlapping area are far enough in the image sequence. That is, given an image (i) of the sequence, the *error variance window* of next image (i+1) will intersect the part of the mosaic image contributed by image (i). Therefore, the loop-detection algorithm keeps a list of the location of the center of every image in the sequence. When an image is added to the mosaic, the system checks if its error variance window is close enough to the center of a sufficiently “old” image. This is one of the parameters which has to be adjusted in the algorithm, preventing the system to stack at continuous crossover detection events. If a crossover is detected, the present image is registered against the mosaic area which is included in the corresponding error variance window. This operation

will normally lead to a new position estimate, and the error variance window will be adjusted according to the variance of the image closer to the present image.

It should be noted that registering two images like image *k* and the area of the mosaic corresponding to image *i* in Figure 6.1 requires the creation of a new *search image*, which is filled with the part of the mosaic that intersects the error variance window of image *k*.

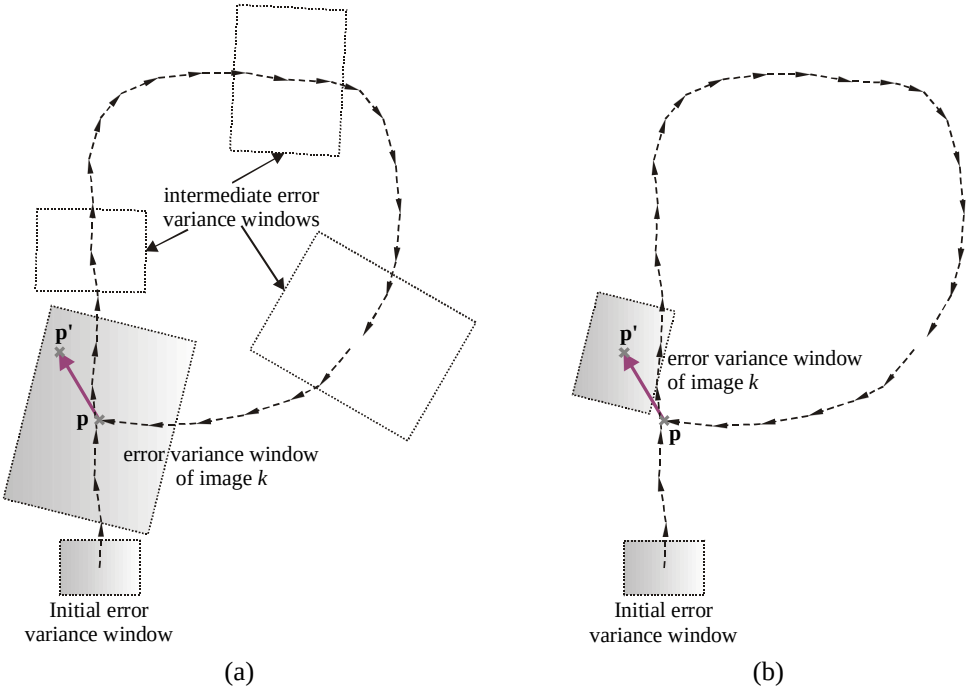


Figure 6.2. Evolution of the error variance windows for every new image. (a) before loop detection and image registration; (b) after registering image *k* against the part of the mosaic corresponding to its error variance window.

6.3 A Kalman filtering approach to smoothing

6.3.1 Introduction

Once the crossover has been detected, the next step consists in re-aligning the sequence of images that form the mosaic, taking into account:

(a) the incremental homographies computed by the mosaicking algorithm (as described in Chapter 5),

(b) the crossover data.

This implies to estimate, at time *k*, the position and orientation of the images prior to time *k*. The estimation of the state of a system $\hat{\mathbf{x}}_i$ at time instant *i*, based on the measures up to time *k*, with $i < k$, is known as **smoothing** in the literature [Jaz70, Bar93]. An optimal estimation technique is needed to minimize the variances on the image positions. MBARI/Stanford researchers perform this optimization through a batch algorithm [Fle97,Fle98]. They re-align the mosaic propagating back the error corrections when a crossing path is detected. In this case, the smoother filter is applied as a batch technique. This approach has the limitation of needing of all the measurements of the image chain to re-align the mosaic, since in a batch algorithm the position estimate of a given image depends on both past and future measurements. On the contrary, we propose to estimate the vehicle motion by means of classical filtering techniques and, at the same time, use them to improve mosaic alignment when crossovers occur [Gar02].

The proposed strategy consists of developing a Kalman filter [Kal60,Kal61] capable of dynamically estimating both the current vehicle position and past trajectory. The state vector $\mathbf{x}(k)$ of our filter includes 3D position (*X*,*Y*,*Z*) and yaw orientation (Ψ) of the submersible. Therefore, the vehicle is assumed to be passively stable in pitch and roll, since its center of mass is below its center of buoyancy.

At every time step, the mosaicking system measures local displacements, which are converted into global position estimates build upon consolidating every image into the global mosaic frame. The measured local displacements are always referenced to a node of the image chain that form the mosaic. The nodes are defined by the location of the central point of an image. Normally, this node is the previous image of the sequence, but eventually it could be another node of the image chain as a consequence of a crossover detection. A general block diagram of the overall smoothing process is illustrated in Figure 6.3. The incremental information provided by the mosaicking system (dashed box at the left of the Figure) feeds the smoother module, which provides a new state estimation. This estimation is used to update the mosaic and vehicle positioning, as well as providing information for detection of future crossovers.

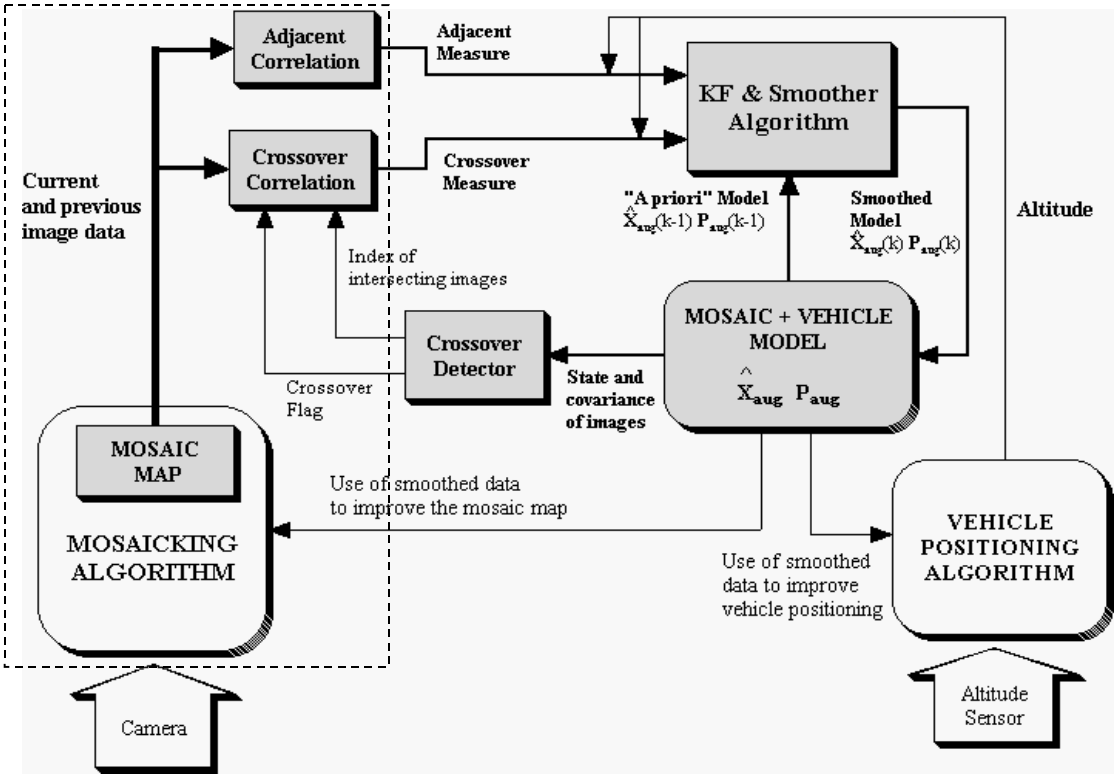


Figure 6.3. Block diagram of the *KF smoother* for global state estimation

Therefore, the state vector of the system has to keep the information regarding the position of the center of all the images of the sequence. Moreover, since measurements are incremental, it is necessary to keep track of all measures. This is not possible with a standard state vector $\mathbf{x}(k)$, since $\mathbf{x}(k)$ contains only information about the current state of the system (at time step *k*). This problem can be solved by augmenting the state vector every time a new measure has to be added to the system. Therefore, the state vector of our filter has the following form [Gar02]:

$$\mathbf{x}_{aug}(k) = \left[\mathbf{x}_v^T(k) \quad \mathbf{x}_{k-1}^T(k) \quad \mathbf{x}_{k-2}^T(k) \quad \ldots \quad \mathbf{x}_0^T(k)\right]^T \tag{6.1}$$

where $\mathbf{x}_v(k)$ is the state of the vehicle and $\{\mathbf{x}_i(k)\ ,\ i = 0,..,k-1\}$ are the locations of the central point of the first *k* images which form the mosaic. When augmenting the state vector, the current estimate covariance error has also to be augmented, to reflect the new state.

$$\mathbf{P}_{aug}(k) = \begin{bmatrix} \mathbf{P}_{v,v}(k) & \mathbf{P}_{v,k-1}(k) & \mathbf{P}_{v,k-2}(k) & \cdots & \mathbf{P}_{v,0}(k) \\ \mathbf{P}_{k-1,v}(k) & \mathbf{P}_{k-1,k-1}(k) & \mathbf{P}_{k-1,k-2}(k) & \cdots & \mathbf{P}_{k-1,0}(k) \\ \mathbf{P}_{k-2,v}(k) & \mathbf{P}_{k-2,k-1}(k) & \mathbf{P}_{k-2,k-2}(k) & \cdots & \mathbf{P}_{k-2,0}(k) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{0,v}(k) & \mathbf{P}_{0,k-1}(k) & \mathbf{P}_{0,k-2}(k) & \cdots & \mathbf{P}_{0,0}(k) \end{bmatrix} \tag{6.2}$$

which is obviously symmetric and where submatrix $\mathbf{P}_{v,v}(k)$ represents the covariance of the vehicle (uncertainty in the vehicle's position at time *k*); and submatrices $\mathbf{P}_{v,j}(k)$ and $\mathbf{P}_{i,j}(k)$ are the covariance between the vehicle and the j^{th} image and the covariance between the i^{th} image and the j^{th} image, respectively.

This leads to the implementation of an *augmented state Kalman filter* (ASKF) [Smi90,Dea99,Ten01], which integrates either the filtering of the vehicle motion and the smoothing of the mosaic based on intersecting trajectories. This approach has several advantages over batch smoothing techniques:

- It is able to integrate all the available information: vehicle's dynamic model, correlation of consecutive (adjacent) images, crossover correlation and other sensor measurements (*e.g.* sonar-based altimeter).
- It continuously updates, with a simple procedure, the state of the vehicle and that of the images which form the mosaic. At the same time, it updates their associated covariances: vehicle-to-vehicle, vehicle-to-image and image-to-image.
- It permits dealing with trajectories of any complexity (*i.e.* multiple loops) in a simple manner. If multiple crossovers occur, the state estimation error covariance $\mathbf{P}_{aug}(k)$ would evolve accordingly to the complexity of the trajectory, due to smoothing in intermediate loops.

Therefore, ASKF is a good framework to keep track of the state of the vehicle and those of every image of the mosaic images; containing all this information in a single state vector.

6.3.2 Constructing the filter

The state estimation $\hat{\mathbf{x}}_{aug}(k)$ and its associated covariance $\mathbf{P}_{aug}(k)$ are **propagated** according to KF *time update* equations:

$$\hat{\mathbf{x}}^{-}_{aug}(k+1) = \mathbf{A}_{aug}(k)\ \hat{\mathbf{x}}_{aug}(k) + \mathbf{B}_{aug}(k)\ \mathbf{u}_{aug}(k) \tag{6.3}$$

$$\mathbf{P}^{-}_{aug}(k+1) = \mathbf{A}_{aug}(k)\ \mathbf{P}_{aug}(k)\ \mathbf{A}^{T}_{aug}(k) + \mathbf{B}_{aug}(k)\ \mathbf{Q}_{aug}(k)\ \mathbf{B}^{T}_{aug}(k) \tag{6.4}$$

where, as the position of images does not vary as a function of time, the system dynamics $\mathbf{A}_{aug}(k)$ and the system noise covariance $\mathbf{Q}_{aug}(k)$ are:

$$\mathbf{A}_{aug}(k) = \begin{bmatrix} \mathbf{A}_{v}(k) & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}; \qquad \mathbf{Q}_{aug}(k) = \begin{bmatrix} \mathbf{Q}_{v}(k) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{6.5}$$

where the identity matrix $\mathbf{I}$ has a size $k \cdot \dim(\mathbf{x}_i)$, and matrices $\mathbf{A}_{v}(k)$ and $\mathbf{Q}_{v}(k)$ will be detailed later in this section.

At every time step *k*, the mosaicking system finds the registration parameters between two consecutive (adjacent) images. Therefore, a new measure $\mathbf{z}_{adj}(k)$ is obtained at every time step. However, when a crossover is detected, an additional measure $\mathbf{z}_{cross}(k)$ is obtained.

In the case of registration of consecutive images (adjacent case), the measure $\mathbf{z}(k)$ measures the position of the k^{th} image (which corresponds to the position of the vehicle) with respect to the $(k-1)^{\text{th}}$ image, so that:

$$\mathbf{z}(k) = \mathbf{z}_{adj}(k) \tag{6.6}$$

$$\mathbf{H}_{aug}(k) = \begin{bmatrix} \mathbf{H}_{v}(k) & -\mathbf{H}_{k-1}(k) & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix} \tag{6.7}$$

However, when a crossover is detected, the current image k^{th} also intersects with the mosaic image. Then, the measurement vector $\mathbf{z}(k)$ becomes:

$$\mathbf{z}(k) = \begin{bmatrix} \mathbf{z}^{T}_{adj}(k) & \mathbf{z}^{T}_{cross}(k) \end{bmatrix}^{T} \tag{6.8}$$

which means that we have two measures, a measure with respect to the previous image $\mathbf{z}_{adj}(k)$, and a second one with respect to the area where the crossover has been detected $\mathbf{z}_{cross}(k)$. If the crossover corresponds to an image *j*, the measurement matrix $\mathbf{H}_{aug}(k)$ incorporates a measurement in column *j*, becoming:

$$\mathbf{H}_{aug}(k) = \begin{matrix} k \\ \\ \end{matrix} \begin{matrix} \quad k-1 \quad & \quad k-2 \quad & \cdots & \quad j+1 \quad & \quad j \quad & \quad j-1 \quad & \cdots & \quad 0 \quad \end{matrix} \begin{bmatrix} \mathbf{H}_v(k) & -\mathbf{H}_{k-1}(k) & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{H}_v(k) & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & -\mathbf{H}_j(k) & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix} \tag{6.9}$$

The form of the measurement matrix of the vehicle $\mathbf{H}_v(k)$, image measurement matrices $\mathbf{H}_{k-1}(k)$ and $\mathbf{H}_j(k)$, and measuring subvectors $\mathbf{z}_{\{adj,\,cross\}}$ are detailed in later in this section.

Then, difference between the measurement $\mathbf{z}(k)$ and the previous "a priory" estimation is given by:

$$\mathbf{r}(k) = \mathbf{z}_{aug}(k) - \mathbf{H}_{aug}(k)\ \hat{\mathbf{x}}^{-}_{aug}(k) \tag{6.10}$$

This difference is called *innovation*, and its covariance $\mathbf{S}(k)$ is:

$$\mathbf{S}(k) = \mathbf{H}_{aug}(k)\ \mathbf{P}^{-}_{aug}(k)\mathbf{H}^{\mathrm{T}}_{aug}(k) + \mathbf{R}(k) \tag{6.11}$$

where $\mathbf{z}_{aug}(k)$ is the measuring vector, $\mathbf{R}(k)$ the measurement error covariance and $\mathbf{H}_{aug}(k)$ is the matrix of the measuring model.

The measures allow the correction of the estimated state and its associated covariance are **corrected** according to the KF *measurement update* equations. So, the filter gain can be expressed as:

$$\mathbf{K}(k) = \mathbf{P}^{-}_{aug}(k)\ \mathbf{H}^{T}_{aug}(k)\ \mathbf{S}(k)^{-1} \tag{6.12}$$

the corrected state estimate corresponds to:

$$\hat{\mathbf{x}}_{aug}(k) = \hat{\mathbf{x}}^{-}_{aug}(k)\ + \mathbf{K}(k)(\mathbf{z}(k) - \mathbf{H}_{aug}(k)\ \hat{\mathbf{x}}^{-}_{aug}(k)) \tag{6.13}$$

and its error covariance becomes:

$$\begin{aligned} \mathbf{P}_{aug}(k) = (\mathbf{I} - \mathbf{K}(k)\mathbf{H}_{aug}(k))\mathbf{P}^{-}_{aug}(k)(\mathbf{I} - \mathbf{K}(k)\mathbf{H}_{aug}(k))^{T} + \\ + \mathbf{K}(k)\mathbf{R}(k)\mathbf{K}(k)^{T} \end{aligned} \tag{6.14}$$

Jazwinski [Jaz70] proved that equation (6.14), which updates the error covariance, is better conditioned for numerical computation than the equivalent usual form shown in equation (6.15) [Kal60].

$$\mathbf{P}_{aug}(k) = \left(\mathbf{I} - \mathbf{K}(k)\mathbf{H}_{aug}(k)\right)\mathbf{P}^{-}_{aug}(k) \tag{6.15}$$

Once the phases of **propagation** and **correction** have been completed, the state and covariance are augmented to add the positioning of the new k^{th} image.

Vehicle model

Although an accurate dynamic model of the vehicle which has been used in simulation is available [Rid01], our KF approach assumes a mathematical description based on a linear model. This assumption is made to obtain a more generic and simple filter, which can be more easily adapted to other submersibles. The vehicle state is described by its position and velocity in the following way:

$$\mathbf{x}_v(k) = \begin{bmatrix} x & y & z & \Psi & \dot{x} & \dot{y} & \dot{z} & \dot{\Psi} \end{bmatrix}^T \tag{6.16}$$

where (x,y) are relative to a mosaic-fixed coordinate system, z is relative to an earth fixed coordinate system and Ψ is the rotated yaw angle in the vehicle fixed coordinate system.

The considered dynamics of the vehicle $\mathbf{A}_v(k)$ assumes a constant velocity model:

$$\mathbf{A}_v(k) = \begin{bmatrix} \mathbf{I}_{4\times4} & dt\cdot\mathbf{I}_{4\times4} \\ \mathbf{0}_{4\times4} & \mathbf{I}_{4\times4} \end{bmatrix} \tag{6.17}$$

where $\mathbf{I}_{4\times4}$ is the 4-dimensional identity, and dt is the sampling period.

Finally, the process noise $\mathbf{Q}_v(k)$ is given by [Ten01]:

$$\mathbf{Q}_v(k) = \begin{bmatrix} \frac{1}{4}dt^4\cdot\sigma_v^2 & \frac{1}{2}dt^3\cdot\sigma_v^2 \\ \frac{1}{2}dt^3\cdot\sigma_v^2 & dt^2\cdot\sigma_v^2 \end{bmatrix} \tag{6.18}$$

where σ_v^2 is the diagonal 4-dimensional matrix of process noise variances in the coordinates (x,y,z,Ψ), used as tuning parameters.

Image model

Every image has an associated state vector which contains the information required to position the corresponding image in the mosaic, so that:

$$\mathbf{x}_i(k) = \begin{bmatrix} x_i & y_i & z_i & \Psi_i \end{bmatrix}^T \quad (i = 0,\ldots,k-1) \tag{6.19}$$

The vectors measuring the displacement with respect to the previous image $\mathbf{z}_{adj}(k)$, and the mosaic area where the crossover has been detected $\mathbf{z}_{cross}(k)$, are described by

$$\mathbf{z}_{\{adj,\,cross\}}(k) = \begin{bmatrix} dx & dy & z & d\Psi \end{bmatrix}^T \tag{6.20}$$

where $(dx,dy,d\Psi)$ are the coordinates of the position of the present image with respect to the previous image ("*adj*" subindex) or with respect to the closer node of the mosaic image ("*cross*" subindex). On the other hand, *z* represents the altitude of the vehicle at the time the present image has been taken. This absolute measurement can be obtained from a sonar altimeter.

Measurement matrix

Looking at equation (6.10) it can be seen that the vehicle measurement matrix $\mathbf{H}_v(k)$ is only used to select the components of the previous "a priori" state estimation. Therefore, it should be defined as

$$\mathbf{H}_v(k) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.21}$$

while image measurement matrices $\mathbf{H}_{k-1}(k)$ and $\mathbf{H}_j(k)$ are

$$\mathbf{H}_{k-1}(k) = \mathbf{H}_j(k) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.22}$$

Note that the component corresponding to measure *z* in equation (6.22) is not updated from the image, but directly provided by the altimeter sensor.

The measurement noise covariance $\mathbf{R}(k)$ may change with each image measurement. It depends on the accuracy of the estimation of the homography transformation **H** described in Chapter 5. Therefore, the *residual error* r_{err} which accounts for the uncertainty in **H** can be computed from [Har00]:

$$r_{err} = \frac{1}{\sqrt{4n}}\left(\sum_{i=1}^{n} d^2\left(\tilde{\mathbf{m}}_i, \mathbf{H}\tilde{\mathbf{m}}_i'\right) + \sum_{i=1}^{n} d^2\left(\tilde{\mathbf{m}}_i', \mathbf{H}^{-1}\tilde{\mathbf{m}}_i\right)\right)^{1/2} \tag{6.23}$$

where n is the number of points which are used to estimate **H**. If n is sufficiently high, the *residual error* r_{err} can be used as an approximation to the unknown *estimation error*. Therefore, **R**(k) can be estimated at every time step from r_{err}. A detailed discussion appears in [Har01]. This residual error is proportional to the standard deviation of the measurements in *x*, *y* and *yaw*. Therefore, it can be used to estimate $\sigma_x(k)$, $\sigma_y(k)$ and $\sigma_\Psi(k)$ by introducing a multiplicative tuning factor.

Then, the measurement covariance matrix in the case of adjacent measure is $\mathbf{R}(k) = \sigma_{adj}^2(k)$. If there is a crossover measurement in addition to the adjacent one, **R**(k) becomes:

$$\mathbf{R}(k) = \begin{bmatrix} \sigma_{adj}^2(k) & \mathbf{0}_{4\times4} \\ \mathbf{0}_{4\times4} & \sigma_{cross}^2(k) \end{bmatrix} \tag{6.24}$$

where $\sigma^2_{\{adj,cross\}}(k) = \begin{bmatrix} \sigma_x^2(k) & 0 & 0 & 0 \\ 0 & \sigma_y^2(k) & 0 & 0 \\ 0 & 0 & \sigma_z^2(k) & 0 \\ 0 & 0 & 0 & \sigma_\Psi^2(k) \end{bmatrix}$.

and σ_x^2, σ_y^2 and σ_Ψ^2 are the measurement variances of image correlation in the mosaic, and σ_z^2 is the variance of the sonar altimeter.

State augmentation

Finally, the information of the present image which is necessary to augment the state vector and its covariance matrix can be obtained from the terms relative to the vehicle. Therefore:

$$\hat{\mathbf{x}}_k(k+1) = \hat{\mathbf{x}}_v(k+1) \tag{6.25}$$

$$\mathbf{P}_{k,(v,k,k-1,\ldots,0)}(k+1) = \mathbf{P}_{v,(v,v,k-1,\ldots,0)}(k+1) \tag{6.26}$$

where equation (6.26) selects the information from the row and column corresponding to the vehicle.

6.4 Optimizations

In order to take advantage of the sequential character of the filter, we should pay attention to its computational cost. Obviously, as the ASKF incorporates new measures at every iteration, its size will increase. Given that it essentially involves matrix multiplications, the cost will be approximately $O(n^3)$, where n is the number of images added to the mosaic. However, in this implementation, the cost can be significantly reduced considering the trivial submatrices (zeros and identities) in $\mathbf{A}_{aug}(k)$, $\mathbf{Q}_{aug}(k)$ and $\mathbf{H}_{aug}(k)$. Then, only the products which involve non-trivial submatrices have to be computed, and then the trivial and non-trivial parts of the matrices can be linked together to form the final matrix. In this way the computational cost can be reduced from $O(n^3)$ to $O(n)$.

Although this improvement is quite significant, as state augments it becomes more and more difficult to obtain real-time performance. Therefore, the number of images which is added to the state should be kept to a minimum. Then, although new images are processed at constant time intervals, incremental position estimations can be injected into the filter only when overlapping between images is below a given threshold, instead of using all the images to update the filter. In this way, the matrices involved in the computations of the ASKF do not increase so rapidly.

6.5 Summary

This chapter completes the description of the visual mosaicking system. A method for the optimal estimation of the position of every image of the sequence after a successful crossover path has been described.

A Kalman filter with augmented state has proved to be the adequate framework for the development the optimal estimator. Although the idea of taking profit of additional information when the vehicle path crosses itself is not new, our approach presents several advantages with respect to the batch system developed by MBARI/ Stanford researchers [Fle96,Fle97]:

- The system is able to cope with several loops.

- It is a sequential algorithm. It can optimize dynamically as new data gets into the system, instead of having to wait for all the data to process it afterwards, like batch filters.
- The filter performs forward iterations which allow the system to estimate the trajectory from the noisy data.
- The measurement noise covariance matrix **R**(*k*) can be updated at every time step depending on the results of the correlation phase. In this way the filter has an idea of how the uncertainty (variance) in image positions evolves.

References

[Bar93] Y. Bar-Shalom and Y.-R. Li, "Estimation and Tracking: Principles, Techniques and Software," Artech House, Boston, 1993.

[Dea99] R.H. Deaves, "Covariance bounds for augmented state Kalman filter application", *IEEE Electronics Letters*, vol. 35, no. 23, pp. 2062-2063, 1999.

[Fle96] S.D. Fleischer, H.H. Wang, S.M. Rock and M.J. Lee, "Video Mosaicking Along Arbitrary Vehicle Paths", in *Proceedings of the OES/IEEE Symposium on Autonomous Underwater Vehicle Technology*, pp. 293-299, 1996.

[Fle97] S.D. Fleischer, S.M. Rock and R.L. Burton, "Global Position Determination and Vehicle Path Estimation from a Vision Sensor for Real-Time Video Mosaicking and Navigation", in *Proceedings of the MTS/IEEE OCEANS 97 Conference*, vol. 1, pp. 641–647, 1997.

[Fle98] S.D. Fleischer and S.M. Rock, "Experimental Validation of a Real-Time Vision Sensor and Navigation System for Intelligent Underwater Vehicles", in *Proceedings of the IEEE Conference on Intelligent Vehicles*, 1998.

[Gar02] R. Garcia, J. Puig and X. Cufi, "Augmented State Kalman Filtering for AUV Navigation", submitted to *IEEE International Conference on Robotics and Automation*, Washington, 2002.

[Har00] R. Hartley and A. Zisserman, "Multiple View Geometry in Computer Vision," Cambridge University Press, 2000.

[Jaz70] A.H. Jazwinski, "Stochastic Processes and Filtering Theory", Academic Press, New York, 1970.

[Kal60] R.E. Kalman, "A new approach to linear filtering and prediction problems", *Transactions of the ASME Journal of Basic Engineering*, pp. 35-45, 1960.

[Kal61] R.E. Kalman and R.S. Bucy, "New results in linear filtering and prediction theory",

Transactions of the ASME Journal of Basic Engineering, pp. 95-108, 1961.

[Rid01] P. Ridao and J. Batlle, "Mathematical Model of an Underwater Robotic Vehicle", in *Proceedings of the IEEE Mediterranean Conference*, Croatia, 2001.

[Smi90] R. Smith, M. Self and P. Cheeseman, "Estimating uncertain spatial relationships in robotics", in I.J. Cox, and G.T. Wilfon (Eds.): *Autonomous Robot Vehicles*, Springer-Verlag, 1990.

[Ten01] I. Tena Ruiz, Y. Petillot, D.M. Lane and C. Salson, "Feature Extraction and Data Association for AUV Concurrent Mapping and Localisation", in *Proceedings of the IEEE International Conference on Robotics and Automation*, Seoul, Korea, pp. 2785-2790, 2001.

Chapter 7

Results

This chapter is devoted to the presentation of some of the results obtained from the techniques described in previous chapters. Different aspects of the work are tested and analyzed. First, an experimental setup to perform laboratory tests and evaluate the accuracy of the mosaic is presented. Then, some of the resulting mosaics of several laboratory sequences are illustrated. Next, the results obtained from real sea trials are shown. Finally, trajectory estimates are re-aligned from information using the crossover data of the mosaic.

7.1 Experimental setup

As the mosaic increases in size, small errors in the motion estimation between consecutive frames provoke an accumulated error. In this work we want to be able to evaluate the nature of error propagation in the resulting mosaic quantitatively. A laboratory setup to obtain an accurate error measurement has been used. The system consists on a robot arm carrying a down-looking camera (see Figure 7.1). This robot has limited accuracy, but good repeatability. The accuracy parameter is defined as the distance between an arbitrarily prescribed location and the one that has actually

been achieved, while the repeatability is measured as the radius of the sphere which contains the points reached after positioning the tool in the same place repeatedly [Sch90].

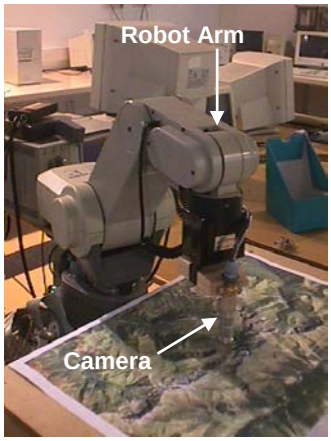


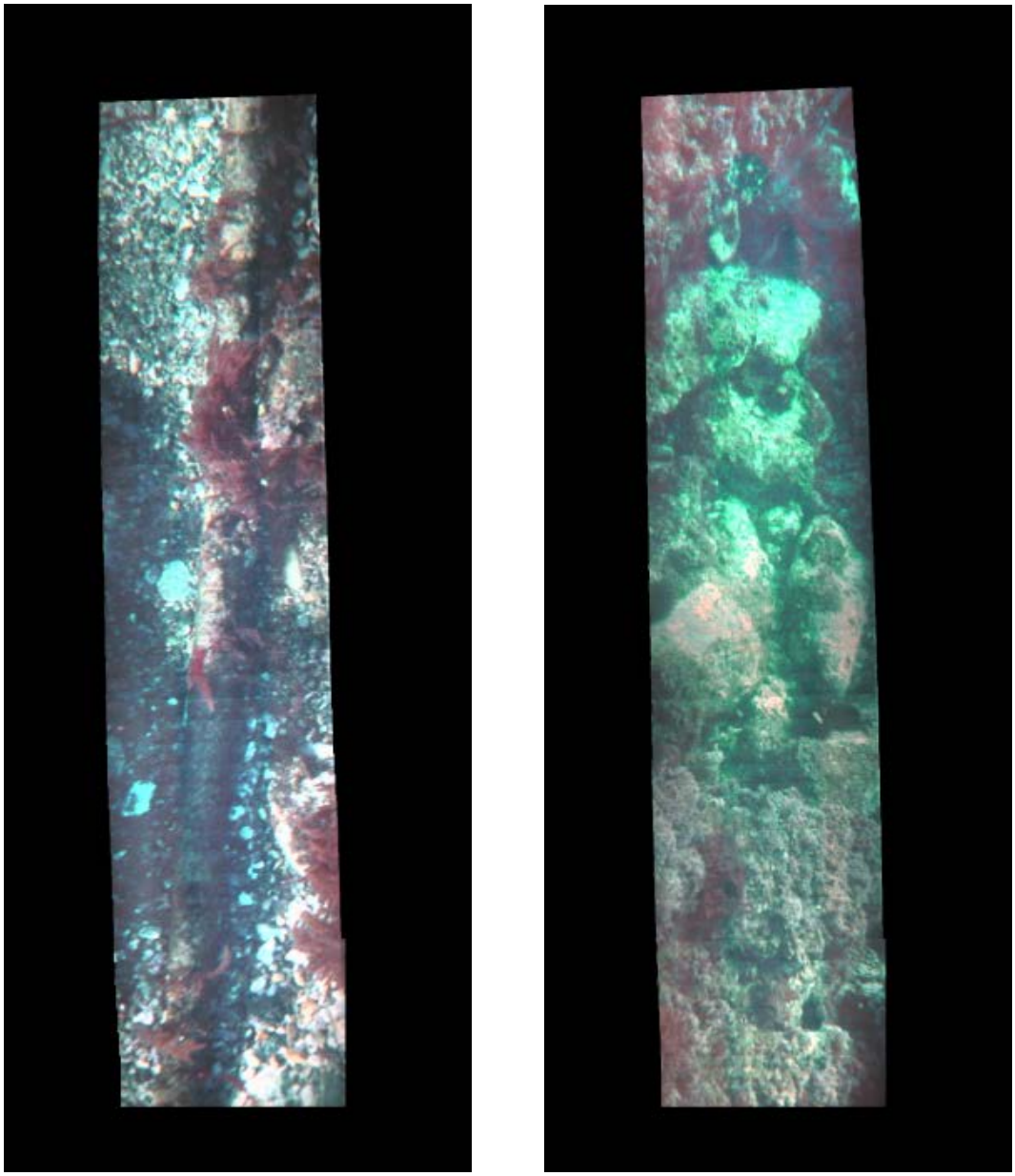
Figure 7.1. Experimental setup. A robot arm carries a down-looking camera and takes images of a poster simulating the sea floor.

The robot arm is required to execute the same pre-defined trajectory twice. A calibration pattern formed by a white background and a matrix of black dots uniformly distributed along the surveyed scene is initially placed under the robot, covering the working area. When the robot executes a trajectory (simulating the motion of a submersible), a sequence of images of the calibration pattern is acquired by the camera. These images will be used to detect the exact image registration parameters. The radial distortion produced by the lenses is corrected for every image. Then, a first motion estimation is obtained from the encoders of the robot, serving as an initial estimate of the actual motion. Next, this estimate is refined by automatically detecting, to subpixel accuracy, the black dots of the calibration pattern in the image sequence. An initial estimate of the position of the black dots in the image is predicted from the information provided by the robot. When these dots are detected in the image the error can be corrected. From the new position of the calibration dots, a 2D projective transform which relates every pixel to the virtual mosaic image is computed. Finally, the pattern is substituted by a poster of the sea floor and the trajectory is executed again. This second time, the acquired image sequence is used as input to the mosaicking system. Since the real trajectory is known, the correctness of the mosaic can be quantified.

7.2 Laboratory tests

The use of the experimental setup enables a quantitative measurement of the accumulated errors of the mosaics created in the lab. We have performed several experiments to compare the accuracy of our mosaicking algorithm against the real values. Normally, as the mosaic increases in size, drift error is expected to increase. However, every time a new image is added to the mosaic, the displacement measures are incremental and referred to the previous image. Therefore, it can be assumed that every new measure is affected by a zero-mean Gaussian error. This means that the trajectory error has a general tendency to increase, but eventually the small misalignment errors can provoke the drift to reduce in certain areas of the path. However, in consecutive images the error may not have a Gaussian distribution with zero mean. This is due to the fact that the matchings detected in the overlapping areas of consecutive images are the same for a few images. Consider for instance a sequence of four images where the first and last images have an overlapping of the 50% of their area. If the most prominent features are detected in this area, the consecutive measurements performed in consecutive images are strongly correlated for a short period of time. And as the sequence gets further from the fourth image, measurement correlation would decrease. Therefore, in the long term, measurements can be considered to be independent, though this is not strictly true in the measurements performed in consecutive overlapping images.

One of the first type of trajectories which has been tested with the experimental setup is the execution of a straight path. Figure 7.2 shows two mosaics constructed from two different image sequences acquired by means of the experimental setup. In both cases a poster of an underwater sequence is placed under the robot. The nature of the images is quite different. The sequence of Figure 7.2(a) shows a pipe surrounded by small stones and some algae. The different depth of the algae and the sea floor provokes a blurring of the image in some areas. The poster of Figure 7.2(b) is a typical underwater scene with big rocks. This image has been taken at more depth, and light turns to the green frequency, decreasing the dynamic range of the image.

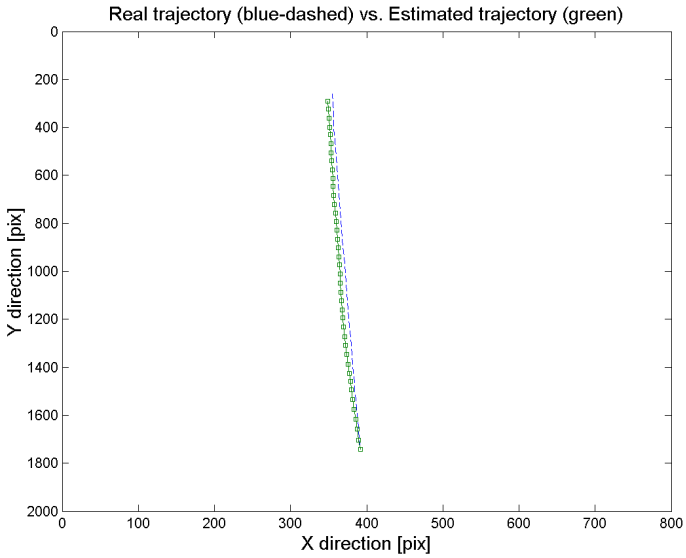


(a) (b)

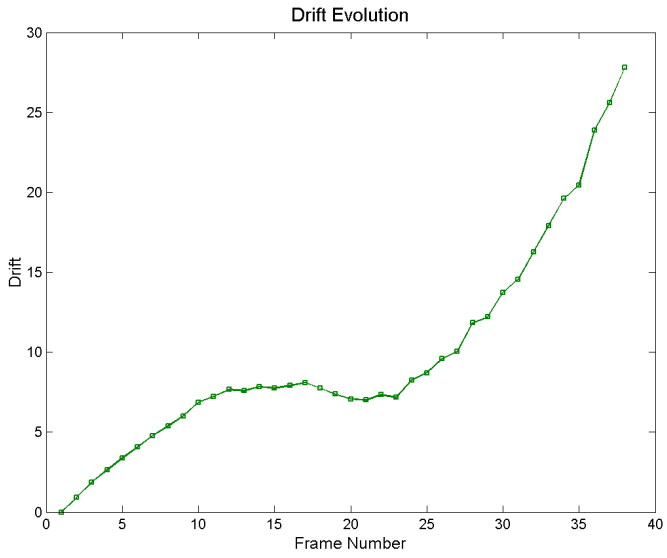
Figure 7.2. Visual mosaics of a straight trajectory generated from the experimental setup. Both mosaics have been created from a sequence of 39 images.

The visual appearance of the generated mosaics is quite good, although small differences in the lighting conditions are observable between images. Real and estimated trajectories are plotted in Figures 7.3(a) and 7.4(a).

The trajectory estimated by the mosaicking system has a drift to the left in the first of the sequences, as illustrated in Figure 7.3(a). The evolution of this drift can be observed in Figure 7.3(b). As expected, drift increases with time, although it is kept nearly constant from images 11 to 17, with a small correction from images 18 to 22.

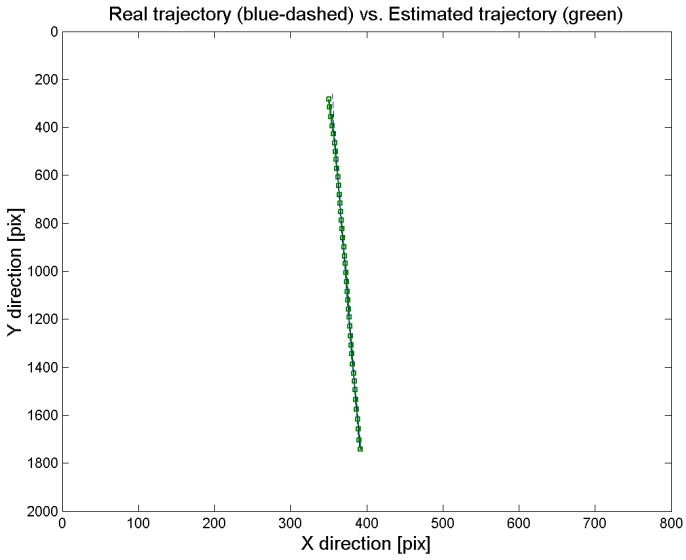


(a)

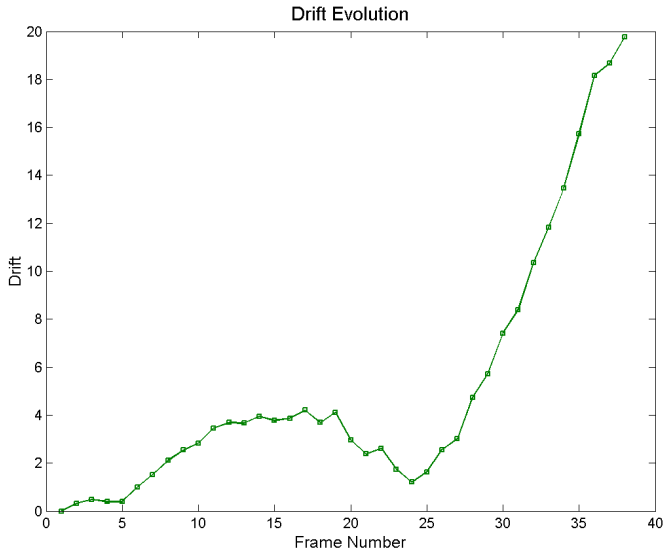


(b)

Figure 7.3. Reconstruction of the straight trajectory computed from the mosaic of Figure 7.2(a).
(a) Real trajectory (dashed-blue) versus estimated trajectory (green with markers).
(b) drift evolution. Final drift is 27 pixels (1.9% of the total trajectory).



(a)



(b)

Figure 7.4. Reconstruction of the straight trajectory computed from the mosaic of Figure 7.2(b). (a) Real trajectory (dashed-blue) versus estimated trajectory (green with markers). (b) Drift evolution. Final drift is 19 pixels (1.3% of the total trajectory).

The trajectory estimated through the mosaic of Figure 7.2(b) follows accurately the real one, although it shortens a the trajectory in the last part of the mosaic, from images 25 up to the end of the path.

Figure 7.5 shows another trajectory followed by the robot in the lab scenario. The scene is a poster showing an aerial photograph of a mountain valley. The trajectory starts at the middle bottom of the image. The size of the sequence is 208 images.

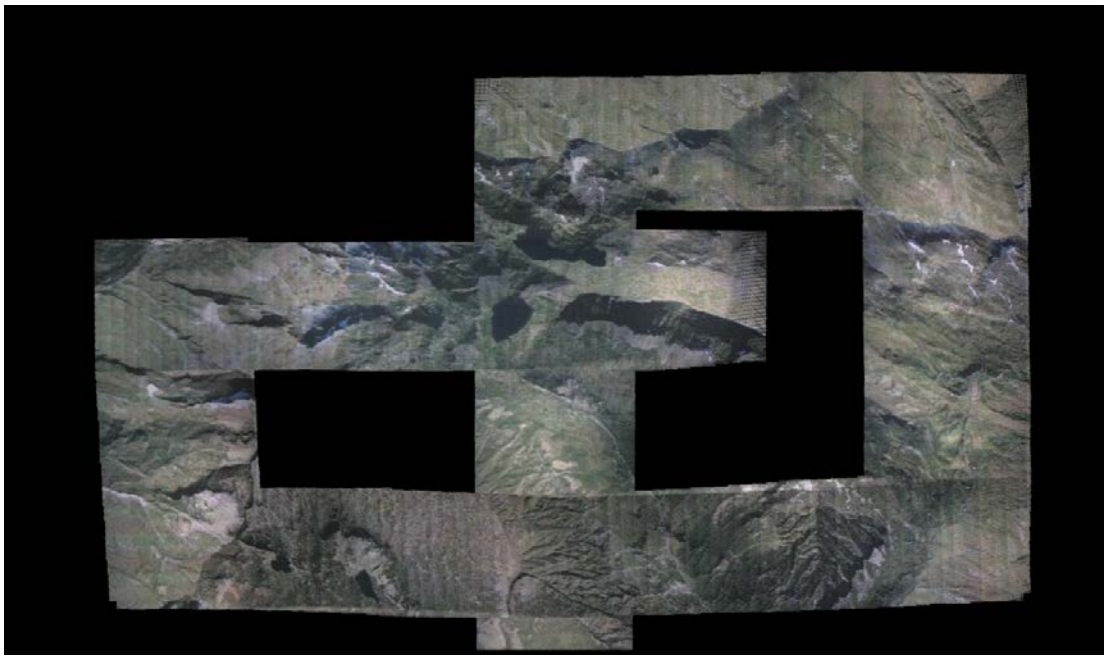


Figure 7.5. Mosaic created from a sequence of 208 images. The size of the mosaic is 2700 × 1600 pixels. The individual images are 384 × 288 pixels

Looking at the mosaic of Figure 7.5, it can be observed that the trajectory described by the camera starts at the lower part of the picture and moves up. If we analyze this mosaic, it appears as “visually correct” within the whole area, except a small misalignment between the first images and the first time the camera crosses an already visited zone, in the lower part of the image. In the second crossover, when the camera passes through its way for a second time, no misalignments are visible. It is not possible to quantify the distortion of the mosaic in any other area of the image by means of a visual inspection. However, if we plot this path against the real one, other drift errors can be detected, as shown in Figure 7.6.

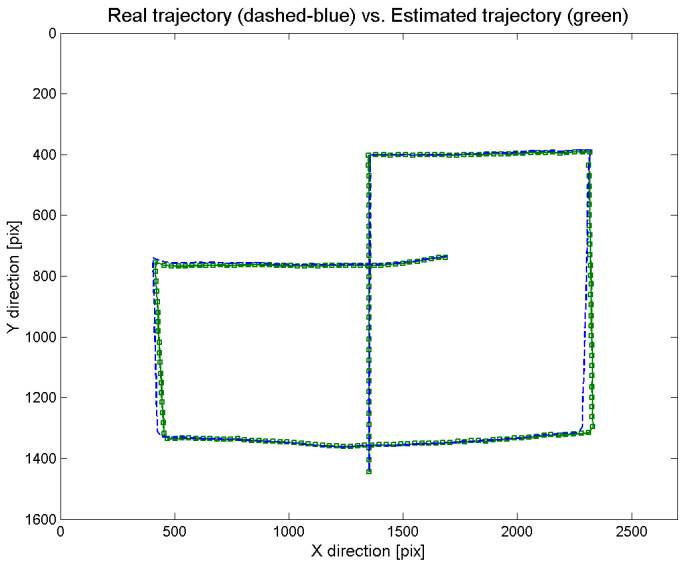
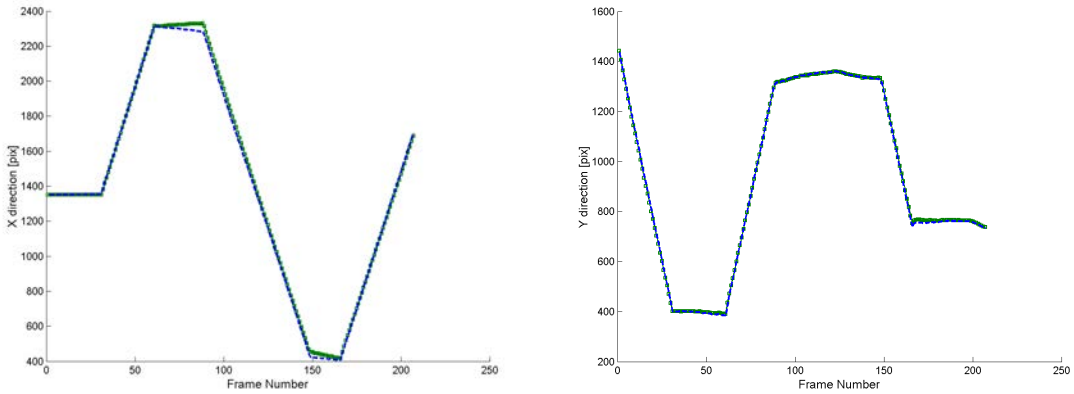


Figure 7.6. Reconstructed trajectory from the mosaic of Figure 7.5. Estimated (dashed) and real (solid) trajectories followed by the robot arm.

The mosaicking algorithm has estimated this trajectory by computing the motion between every pair of consecutive images in the sequence. Figure 7.5 represents the temporal evolution of the trajectory followed by the pixel located at the center of every image. Since there exists a considerable overlap between every image and the next one, the same area is common to several images. In this case, an alternative would be the selection of a reference image, and the estimation of motion between this image and the next few frames. This second approach yields to a mosaic presenting less drift (because incremental errors increase more slowly), but usually with a worst visual appearance. On the contrary, the methodology that has been used to create the mosaic of Figure 7.5 generates maps with a good visual appearance, but is less adequate to estimate the position of an underwater vehicle.

The lower right corner of Figure 7.4 presents a drift of 46 pixels. If we want to use the mosaicking system to position an underwater vehicle, this drift would suppose an error in the order of 25 to 50 cm depending on the altitude of navigation of the submersible. This drift is kept approximately constant along the horizontal path in the lower part of the figure. This can be more easily observed in Figure 7.5. Almost at the end of the path (in the second crossover), the drift has been reduced to 6 pixels. In this case, local errors have produced a reduction of the drift. Then, the visual

analysis of the mosaic looks nearly perfect, while errors in the estimation of the trajectory have occurred. Figure 7.8 shows the peculiar evolution of the accumulated drift in this sequence. The maximum error is 47 pixels (a 0.7% of the total path), while the final error is 15 pixels, which represents only a 0.2% of the total trajectory. This is a clear example of drift reduction from accumulative errors.



(a) (b)

Figure 7.7. Temporal evolution of the estimated (green) and real (blue-dashed) trajectories for the X and Y coordinates, respectively (a) and (b), for the mosaic of Figure 7.5.

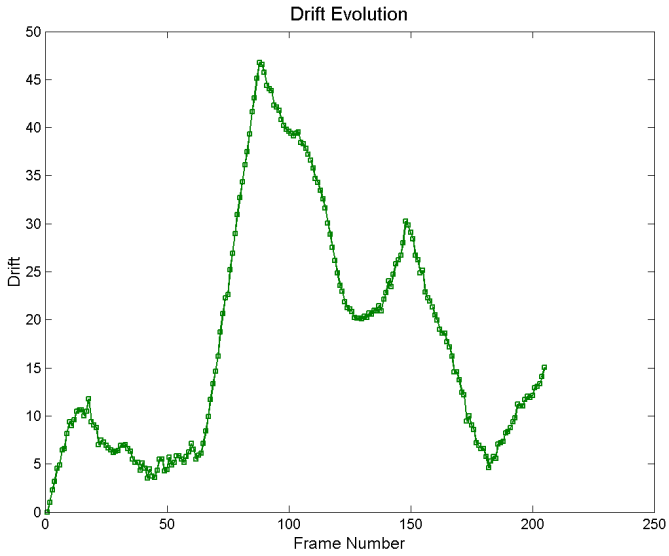


Figure 7.8. Drift accumulated by the mosaic of Figure 7.5.

Figure 7.9 shows an oval trajectory. The robot starts and finishes at the center left hand side of the image. It can be observed that the final position of the camera is quite close to the original one. However, a small misalignment can be observed in this area, while in the rest of the image sequence small accumulative errors are not perceptible (although they occur).

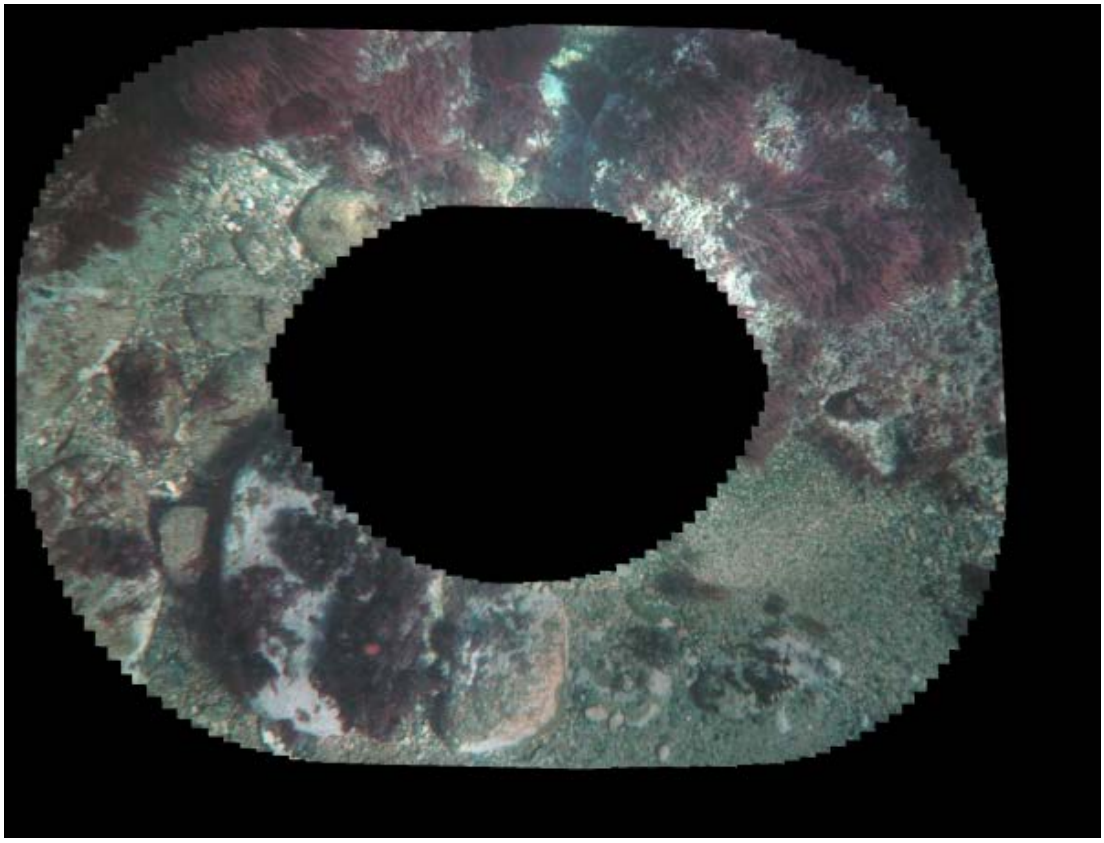


Figure 7.9. Mosaic created from a sequence of 130 images. It starts and ends at the center left of the image. A small misalignment of 37 pixels can be observed.

Figures 7.10 and 7.11 show the reconstructed trajectory and drift evolution of the mosaic of Figure 7.9. Again the maximum drift occurs at the end of the path, being lower than a 1% of the total trajectory. It can be observed from images 76 to 92 the accumulative error produce a reduction of drift. However, from the image 93 up to the end of the path drift increases quite seriously. This proves that in the long term the measurements are uncorrelated, but there is a strong correlation in certain areas, like in the last part of this oval trajectory.

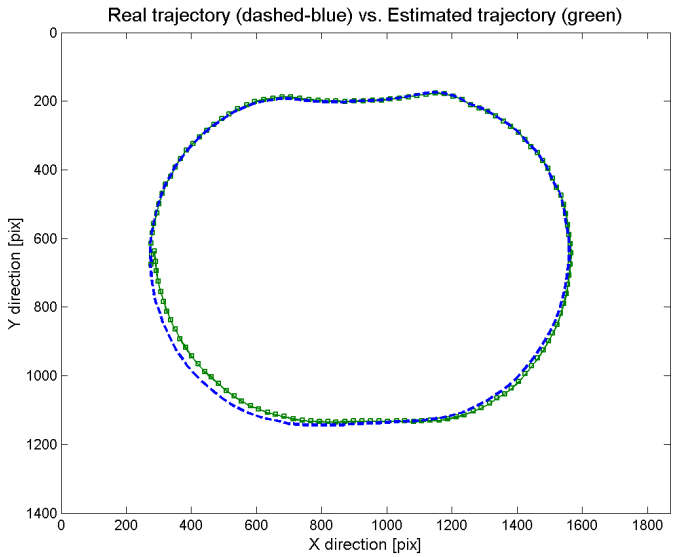


Figure 7.10. Reconstructed trajectory from the mosaic of Figure 7.9. Estimated (green with markers) and real (dashed-blue) trajectories followed by the robot arm.

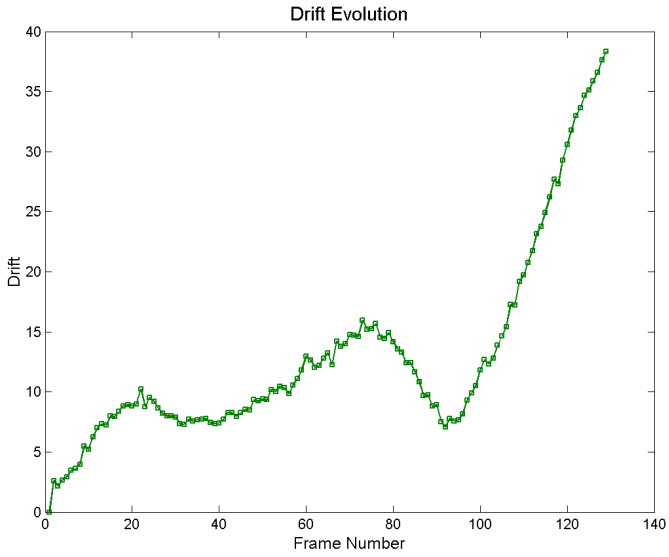
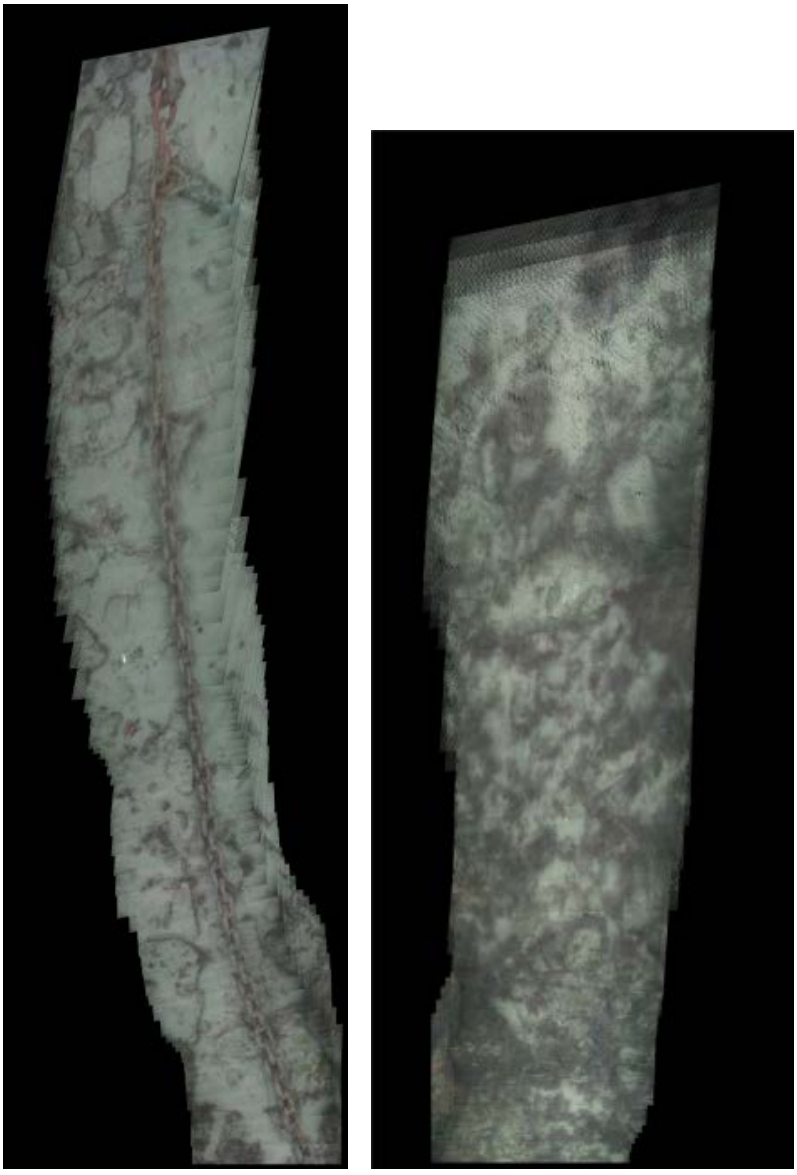


Figure 7.11. Drift evolution of the trajectory of Figure 7.9. The final drift of 37 pixels supposes an error of a 1% of the total path.

7.3 Sea trials

After the encouraging results obtained from the previous section, several experiments have been performed in the sea. All the field tests have been performed in coastal

waters of Costa Brava, with the underwater vehicle URIS in teleoperated mode. To perform each experimental run, the pilot teleoperates the vehicle positioning it at suitable range above the seabed. Then, as the vehicle moves, the acquired images are sent to the surface through the umbilical tether, where they are either stored to a tape or processed in real time. The next figures show some of the constructed mosaics from the sea trials that took place during the months of June and July, 2001.



(a) (b)

Figure 7.12. Two sample trajectories followed by the submersible during a sea trial.

Figure 7.12(a) shows the mosaic constructed while the vehicle was following a submersed chain. At the beginning of the sequence the chain is at the range of the sea bed, but, as the image sequence progresses, it goes up from the floor. It is possible to see how the underlying assumption of flat scene is violated. However, the vehicle path can be reconstructed from the mosaic without a major problem. Although point correspondences detected in the chain are correctly established, the points detected in the chain undergo a different apparent motion than the background sea floor points, due to differences in range. For this reason the LMedS algorithm detects them as outliers. Unfortunately, it is not possible to quantify the errors which are produced in real sea trials, since the real trajectory cannot be recovered from any other sensors.

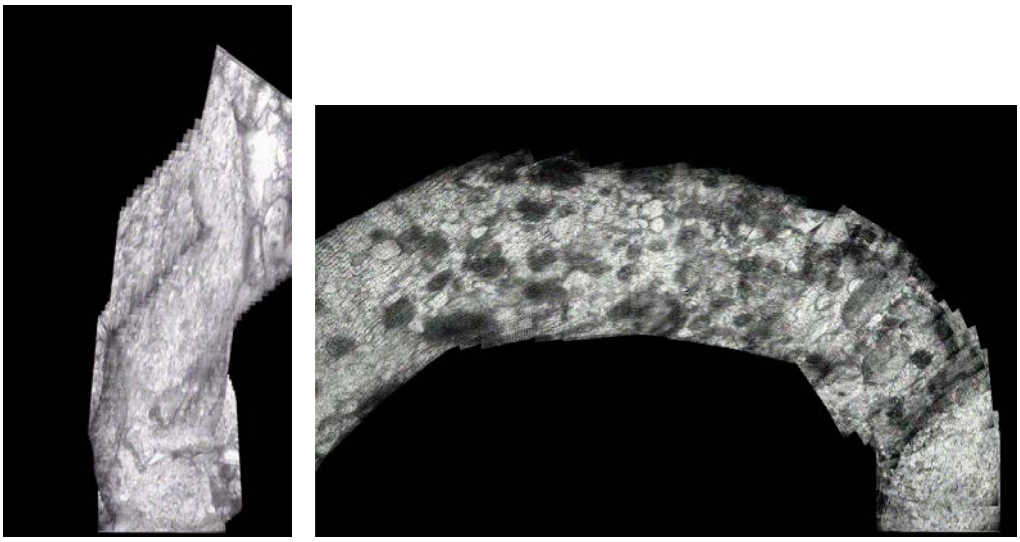


Figure 7.13. Sample trajectories described by URIS at sea.

7.4 Considering crossover information

The KF approach to realign the mosaic has been tested with synthetic and real data. In order to create a set of test trajectories, the *Autonomous Underwater Vehicle Simulator* (AUVS) for virtual and/or real applications [Rid01a,Rid01b] has been used. This simulator incorporates the identified dynamic model of GARBI, and it is able to create the sort of trajectories that the vehicle would follow in a real mission. Figures 7.15–7.18 show different crossover trajectories generated by means of the AUVS. The correspondences in the intersecting path after crossover detection are

illustrated with a thin dashed line. First, the trajectory without noise (dashed-blue) is generated. Then, a trajectory simulating the robot drift is generated by adding accumulative zero-mean Gaussian noise (green) to the real one. Finally, the KF computes the smoothed trajectory (magenta).

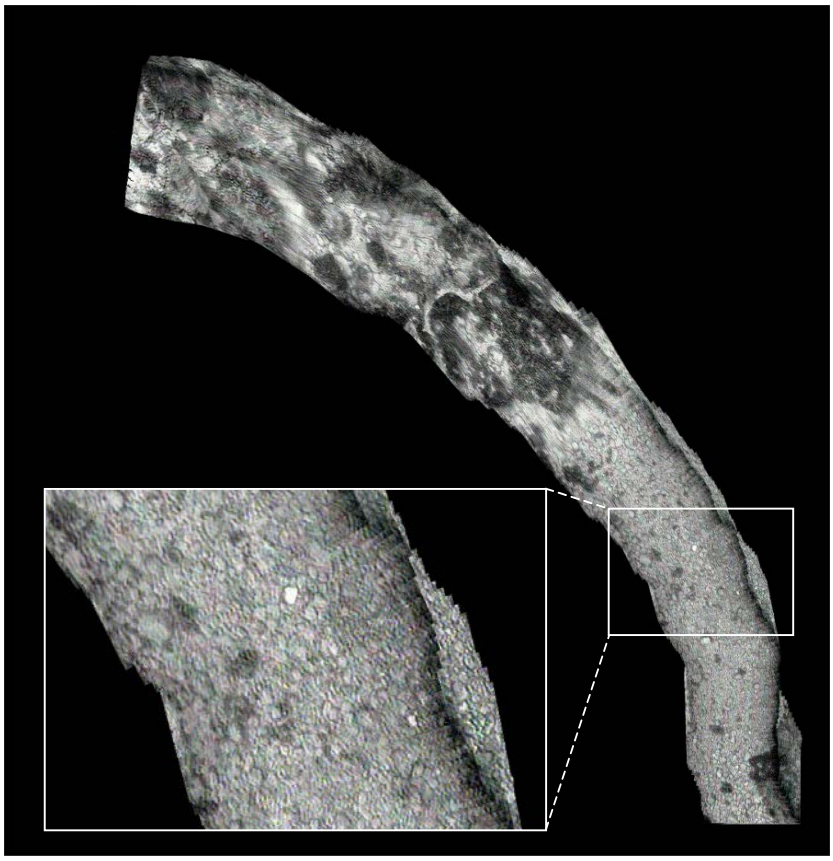
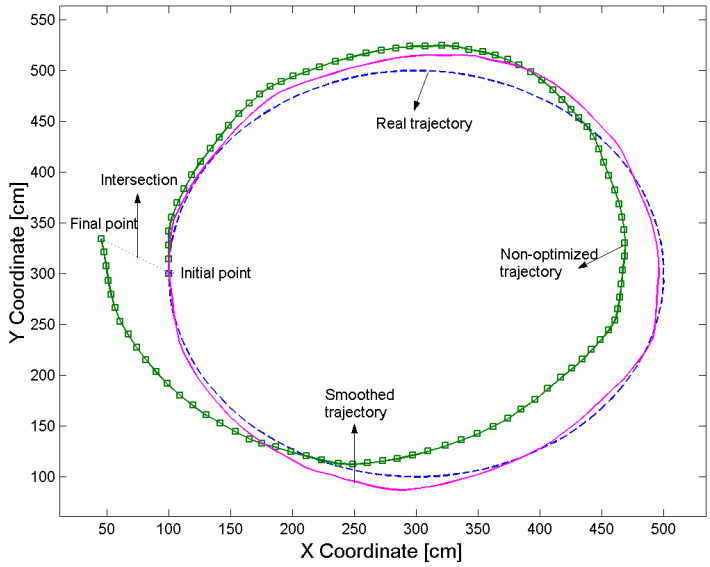
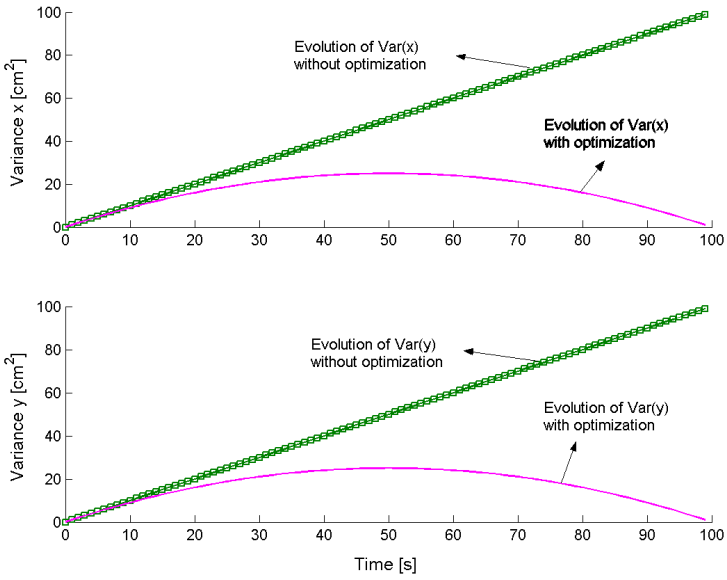


Figure 7.14. Resulting mosaic constructed during a real mission. The zoom shows the noisy nature of the images and the lighting effect produced by the camera at the bottom left hand side of every frame.

Figures 7.15(a) and 7.16(a) show a circular and square path, respectively. Their (b) counterparts illustrate the evolution of the variance of the *X* and *Y* components before (green) and after (magenta) smoothing. It can be observed that the variance associated to vehicle position increases indefinitely as the vehicle moves. However, when the vehicle detects a crossing path, variance is reduced to the previously existing variance at the intersection node.

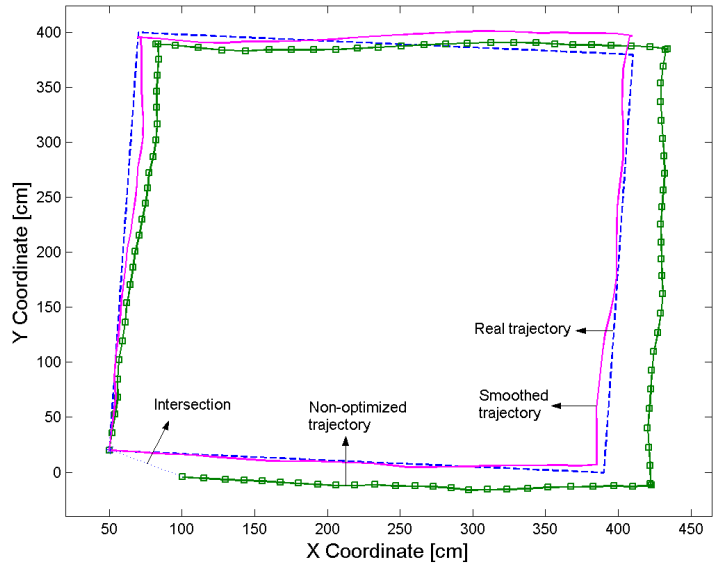


(a)

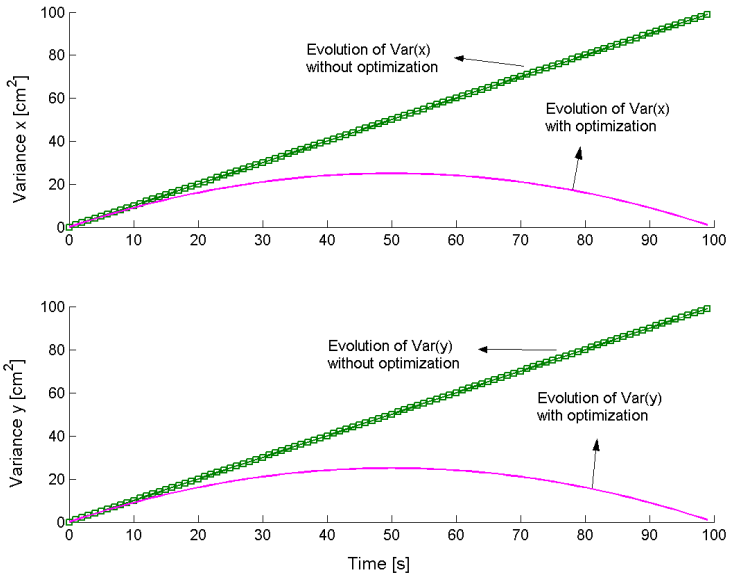


(b)

Figure 7.15. Crossover simulation. (a) The graphic shows the trajectory without noise (dashed-blue), trajectory with accumulative Gaussian noise (green) and smoothed trajectory (magenta) after applying the ASKF. (b) Variance evolution in X and Y before (black) and after (magenta) smoothing.

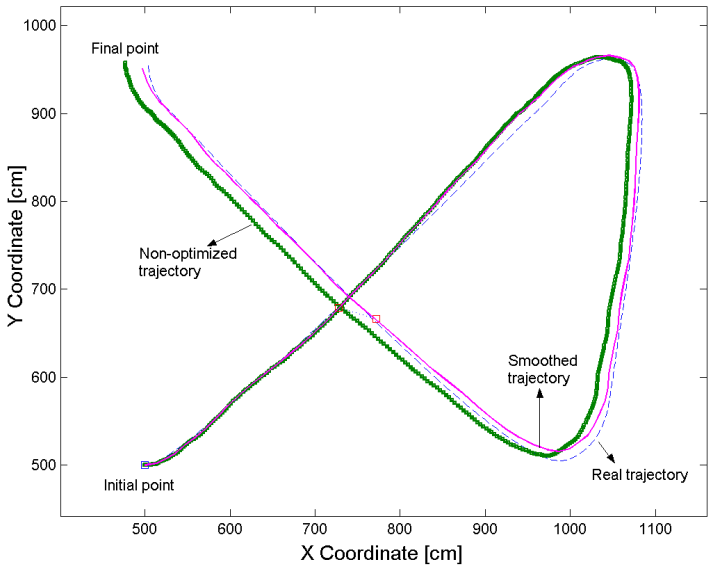


(a)

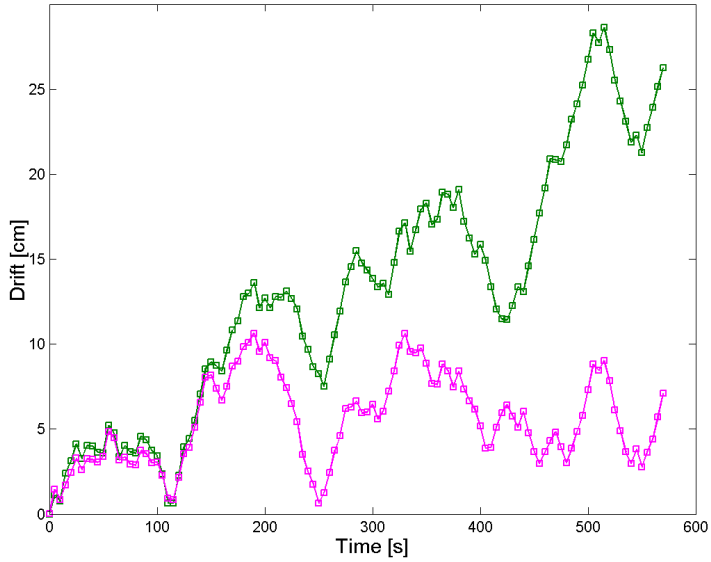


(b)

Figure 7.16. Square trajectory simulating a crossover. (a) The filter re-adjusts the estimated trajectory after smoothing (magenta).The real trajectory is drawn in dashed-blue, and the simulation of the trajectory computed by the mosaicking system in green. Every small square represents a measurement point (image node) (b)Variance evolution before (green) and after (magenta) smoothing.



(a)



(b)

Figure 7.17. Crossover simulation. (a) Trajectory without noise (dashed-blue), trajectory with accumulative Gaussian noise (green) and smoothed trajectory (magenta); (b) Drift evolution: before (green) and after (magenta) smoothing.

It should be noted that the evolution of the variance (uncertainty in the image exact location) does not have a direct relation with drift error (difference between the

exact path and the estimated one). We can observe from the preceding images that the variance always increases linearly before smoothing (black), while after smoothing (magenta) error variance decreases as the position of the images is closer to the crossover point.

In order to test the capabilities of the ASKF to deal with several loops, we have also used the AUVS to generate a trajectory with 3 crossing paths. In this experiment the vehicle navigates in an area of 50×50 meters. Then, this trajectory has been scaled and executed by the robot arm of our experimental setup, as described in section 7.1. A poster of the sea floor is again placed under the robot, covering the working area. The robot executes the trajectory (simulating the motion of a submersible) and the camera acquires the image sequence which will be processed by the mosaicking system. In this way the real and estimated trajectory followed by the robot are known [Gar01]. Figure 7.18 shows the sample trajectory with 3 crossover paths. The dashed blue line represents the real trajectory and the trajectory estimated by the mosaicking system is shown in solid green. The path of the vehicle starts at the at the bottom left hand side of the figure. The evolution of the smoothed trajectory can be followed in the different sub-figures (drawn in magenta with a marker at every sample point). Figure 7.18(a) shows the trajectory filtered by the ASKF before the first crossover is detected. It basically follows the trajectory computed by the mosaicking system (solid-green). It can be observed that the smoothed trajectory stops before intersecting its path. This means that the *crossover detector* tells the robot that it has already arrived to a crossover point, although the robot thought that it was further from that point. With this information, the ASKF smooths back the positions of the previous images, and then it goes on filtering the vehicle trajectory (Figure 7.18(b)). Again, an intersection is detected. It can be observed in Figure 7.18(c) that the smoothed trajectory in the top of the map goes down, after the second crossover, approaching the real trajectory. Finally, Figure 7.18(d) shows the smoothed trajectory after the third path intersection. Here we have shown the evolution of the smoothed trajectory superimposed on the final real and measured paths; however, it should be noted that the smoothed trajectory is updated sequentially, as the mosaic provides every new measurement.

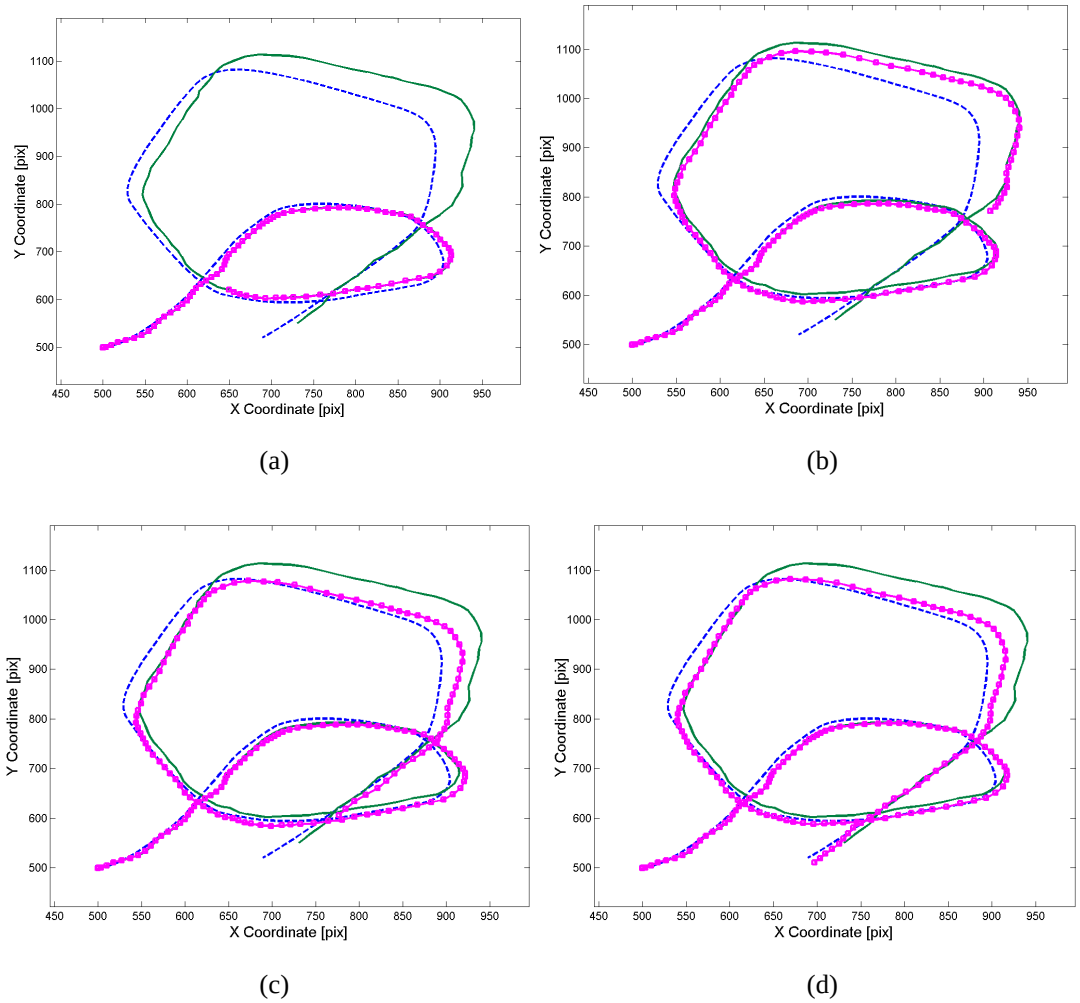


Figure 7.18. Sample trajectory with 3 intersections. The sequence shows the real trajectory (*dashed-blue*), non-smoothed estimated trajectory (*solid-green*) and the evolution of the smoothed trajectory (*magenta with markers*) as new crossovers are detected.

The accumulated drift is illustrated in Figure 7.19. Initially, both the trajectory computed by the mosaic and the ASKF smoother present a similar drift. When the first crossover is detected, drift of the smoothed path can be reduced to a very low value. Then it increases again, but the second loop keeps the drift in a considerably smaller range than the measured trajectory. Figure 7.20 shows the drift independently plotted for the *x* and *y* coordinates. The diagonal components of the final state error covariance $\mathbf{P}_{aug}$ are used to compute the uncertainty bounds of the smoothed trajectory, drawn as 3 times the estimated standard deviation at every point.

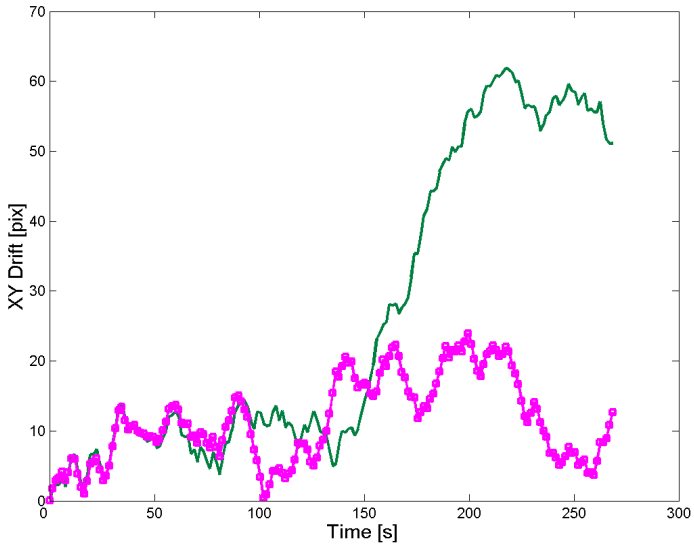


Figure 7.19. Drift evolution of the vehicle trajectory in the mosaic plane. Drift of the smoothed (*magenta with markers*) and non-smoothed (*solid-green*) trajectories.

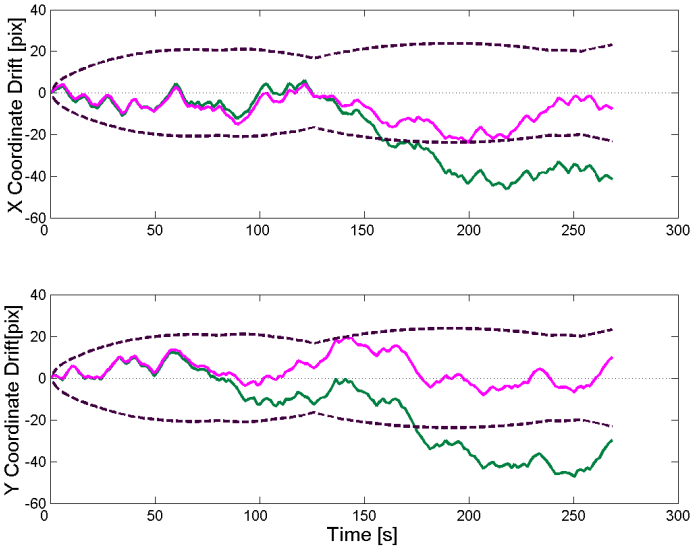


Figure 7.20. Drift of the smoothed (*magenta*) and non-smoothed (*solid-green*) trajectories for the X and Y coordinates. The uncertainty bounds of the smoothed trajectory (*dashed*) is drawn as 3 times the estimated std. deviation (obtained from the final state error covariance matrix $\mathbf{P}_{aug}$).

Finally, Figure 7.21 shows the result of applying smoothing to the oval trajectory of Figure 7.9. The misalignment at the center left hand side of Figure 7.9 is not perceptible in the smoothed version.

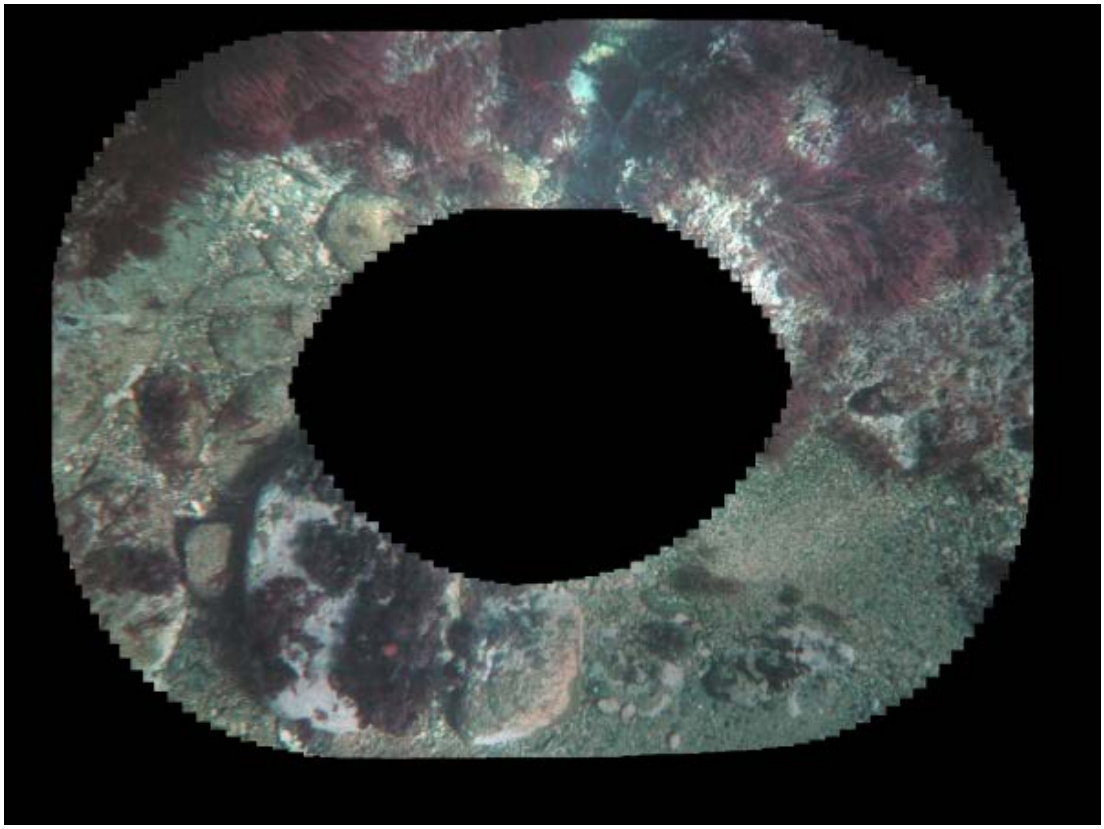


Figure 7.21. Resulting mosaic image after applying smoothing to the image of Figure 7.9. The misalignment at the center left hand side does not exist.

7.5 Analyzing the efficiency of the system

At this juncture, it has become clear that the mosaicking system should serve as a positioning tool to navigate close to the ocean floor. Therefore, the efficiency of the algorithm is a key factor to allow the system to work in real time. The mosaicking system has been coded in C++. Since our underwater vehicles are relatively slow (typical speeds within a mission are below 0.5 m/s), a cycle time around 1 second should be enough to achieve real-time performance. An adequate parametrization of the system allows this performance to be reached without need of specific hardware.

Table 7.1 shows the timing of the different phases of the algorithm. Tests have been performed with an AMD-K7 1300 MHz, and the system has been parametrized with the configuration used to create the mosaics of Figure 7.12. This configuration is detailed in Table 7.2.

Table 7.1. Timing of the different phases of the mosaicking algorithm.

Phase	*Time (in seconds)*	*Accumulated Time*
Detection of Interest points	0.170	
Textural-aided Correlation	0.370	0.540
Outlier Rejection	0.340	0.880
Motion Estimation	0.060	0.940
Mosaic Construction	0.210	1.150

It can be observed from Table 7.1 that motion estimation parameters are available after 940 ms. This means that the position of the vehicle can be updated every 940 ms. However, copying the present image to the mosaic frame takes 210 additional milliseconds. Therefore, if we want to use the system just as a positioning sensor, the cycle can be completed in less than a second, and if our aim is to construct a visual map, the cycle goes up to 1.15 seconds, which can still be considered real time performance.

Table 7.2. System parametrization.

Module	**Parameter**	**Value**
-	Image size	192×144
Interest-point detector	Number of Interest Points	25
Correspondences	Search Window	35×35
	Correlation Window	17×17
	Subsampling factor	4
LMedS	Iterations	100
Motion Estimation	Motion Model	Affine

It can be observed from the parameters of Table 7.2 that the mosaicking algorithm is searching for correspondences within a 35×35 search window. If the apparent motion of the vehicle is too high, the system would fail to find the correspondences in this area. A possible alternative would be to use a multi-resolution approach,

keeping the size of the search window, but subsampling the image to half its size. This solution has not been implemented in the present work but it will be included in the future improvements.

7.6 Summary

We have presented in this Chapter some of the results which have been obtained either from simulation, lab tests and real sea trials. From the set of tests which have been performed it is obvious that as the mosaic increases in size, drift has a clear tendency to augment in the long term. Unfortunately, it is not possible evaluate quantitatively the accuracy of the resulting mosaics and their associated trajectory estimation in real sea trials. However, the lab results obtained through an experimental setup show that the errors in the estimation of the vehicle trajectory in lab conditions may be below a 3% of the total trajectory. Nevertheless, meeting lab conditions may be too difficult to achieve in real applications, where lighting problems, scattering, bad visibility, etc. could deteriorate the signal-to-noise ratio of lab images. For this reason, the possibility of exploiting the additional information gained at crossover paths opens appears as an interesting alternative for applying visual mosaicking to the *Concurrent Mapping and Localization* problem.

Summarizing, the quality of the obtained mosaics encourages the use of vision as a positioning sensor for small AUVs, when these submersible navigate close to the sea floor.

References

[Gar01] R. Garcia, J. Batlle and X. Cufi, “A System to Evaluate the Accuracy of a Visual Mosaicking Methodology,” in *Proc. of the MTS/IEEE OCEANS Conf.*, Hawaii, 2001, *in press*.

[Rid01a] P. Ridao, J. Batlle and M. Carreras, “An underwater autonomous agent: from simulation to experimentation”, in *Proceedings of the IEEE Mediterranean Conference on Control and Automation*, Croatia, 2001.

[Rid01b] P. Ridao, “A hybrid control architecture for an AUV”, Ph.D. Thesis, University of Girona, 2001.

[Sch90] R.J. Schilling, *Fundamentals of Robotics: Analysis and Control*, Prentice-Hall International, 1990.

Chapter 8

Concluding Remarks

This chapter summarizes the conclusions of the presented work. This dissertation has basically proposed a method to construct visual mosaics of the ocean floor at the same time that this information is used to estimate the position of an underwater vehicle. The application of computer vision to underwater navigation has proved to be an effective tool when the media conditions are adequate.

8.1 Summary

One of the main interest points of this work is whether it is possible to make use of computer vision in automatic UUV positioning and navigation. The answer is not only affirmative, but strictly necessary for small low-cost vehicles where other sensors are very limited due to both their size and price.

Therefore, the construction of visual mosaics of the ocean floor can provide accurate position estimates for local navigation of underwater vehicles. We have presented an approach to quantify the distortion across visual mosaics. A solution to the problem of measuring error propagation in the construction of a mosaic has been

proposed. This solution is valid for laboratory testing only, but it has also proved to be helpful in testing and tuning the different parameters of our mosaicking system, *e.g.* selection of texture operators, number of interest points, comparative of different motion models, etc.

The construction of a mosaic for robot navigation suffers from drift errors, which are not distributed uniformly across the mosaic, but rather appear in certain areas. This is probably due to the lack of adequate information in the images acquired on these areas to produce good motion estimates.

Our approach to construct underwater mosaics has been validated by means of the experimental set-up. Further experiments will be carried out to enhance the performance of the mosaicking algorithm. In this work we have only dealt with planar scenes. In the future, perspective projection of non-planar objects should be studied in the construction of mosaics. The use of this validation tool will suppose a step forward towards the creation of a robust mosaicking methodology to be applied in real missions.

Finally, a method for the optimal estimation of the position of every image of the sequence after a successful crossover path has been described. The Kalman filter with augmented state (ASKF) approach has proved to be the adequate framework for the development the optimal estimator. Although the idea of taking profit of additional information when the vehicle path crosses itself is not new, our approach presents several advantages with respect to batch systems. First, the system is able to cope with several loops. Second, it is a sequential algorithm. Therefore it can optimize dynamically as new data gets into the system, instead of having to wait for all the data to process it afterwards, like batch filters. Third, the filter performs forward iterations which allow the system to estimate the trajectory from the noisy data. Finally, the measurement noise covariance matrix $\mathbf{R}(k)$ can be updated at every time step depending on the residual error measured from the images. In this way the filter has an idea of how the uncertainty (variance) in image positions evolves.

8.2 Contributions of this research

The contributions of this dissertation can be divided in four blocks:

- Comparative study of the *state-of-the-art* of mosaicking systems for underwater navigation.
- Proposal of a method to solve the correspondence problem by exploiting the textural characteristics of the selected features. An accurate use of the texture information has proved to improve to a large extent the quality of the detected features.
- Proposal of a new algorithm to construct mosaics of the ocean floor. The computational cost of the method allows its use in real missions for the navigation of underwater vehicles.
- Proposal of a smoothing strategy, that can be integrated in the mosaicking system, which reduces the uncertainty of the vehicle location when the trajectory of the submersible crosses itself.

8.3 Future work

The accuracy of the mosaicking systems is limited due to several factors. Improvement on the reliability on any of these factors implies an improvement in the accuracy of the whole system. Feature-based methods suffer from an inherent uncertainty in the measurement of the image features. The impossibility of measuring the exact position of the features in the image causes errors in the motion estimation process and, therefore, in the mosaic alignment. However, improvement could be obtained if all the incoming images are correlated directly with the same base image, instead of incrementally registering consecutive images. When the intersection between the present image and the base frame is smaller than a given threshold, a new base frame has to be selected. In this way, accumulative error would propagate slower through the image chain.

The work presented in this dissertation proves that the incorporation of texture information can improve to a large extent the detection of feature correspondences in consecutive images. For this reason, the system is opened to incorporate in the future other texture operators which could be of interest in solving the correspondence problem.

Fortunately, robust algorithms such as LMedS or RANSAC can, to a large extent, reduce the amount of anomalous data (so-called “outliers”). The development of new algorithms that could effectively detect non-consistent data would improve the results of mosaicking systems, as well as finding more accurate and reliable feature detectors.

On the other hand, featureless methods face the problem of non-linear Least Squares estimation. At the moment, these methods require a good initial guess at the solution in order to converge to the global minimum. Otherwise the iteration may lead to a local minimum, or may not converge at all.

Most of the mosaicking systems described in the scientific literature compute a planar transformation to register the images. Some other systems estimate 3D motion with respect to a sea floor which is assumed to be planar. However the mosaicked area may not only be non-planar, but may also present high variations in depth. In this case, new motion estimation and/or 3D structure recovering algorithms should be developed to gain a global perspective of the surveyed zone. This is relevant in the case of the exploration of wrecks or submersed structures with a considerable 3D shape.

On the other hand, all the featureless methods, as well as some of the feature-based, minimize a cost function that depends uniquely on the image intensity values and (in some cases) variations in the radiosity of the image. Future research could improve image registration by adding new parameters to the cost function, such as textural characteristics, leading to more robust estimates.

Therefore, visual mosaics can be considered as an important tool that allows accurate motion estimation and positioning of underwater vehicles. Although this methodology has still serious limitations (*e.g.* limited surveyed areas, drift errors, etc), the increasing computer power will allow the development of new real-time mosaicking algorithms, thus providing more accurate estimates and increasing the mosaic sizes in next few years. Moreover, most of the visual mosaicking systems that have been analyzed are uniquely taking information from the on-board cameras. In some cases, other on-board sensors such as compasses, Doppler sonar or inertial navigation systems (INS) have been timidly used. It has been proved that local accuracy provided by visual sensing is higher than that of any other sensor (with a similar cost). However, as the mosaic increases in size, a considerable drift can bias

the system. For this reason, sensor fusion integrating vision with other sensors that are not subject to drift, such as compass sensors or some LBL sonar that are able to provide absolute position readings, could considerably improve the accuracy of the construction of a visual map, and, therefore, improve navigation.

8.4 Related publications

The publications listed below are direct consequence of the evolution of this dissertation in the last three years:

[1] R. Garcia, S. Negahdaripour and X. Cufi, “A Review on Mosaicking Systems for Underwater Applications”, to be submitted to the *IEEE Journal on Oceanic Engineering*.

[2] R. Garcia, J. Puig and X. Cufi, “Augmented State Kalman Filtering for AUV Navigation”, submitted to *IEEE International Conference on Robotics and Automation* (ICRA-2002), Washington, 2002.

[3] R. Garcia, J. Batlle and X. Cufi, “A System to Evaluate the Accuracy of a Visual Mosaicking Methodology”, *MTS/IEEE OCEANS Conference*, Hawaii, 2001, *to be published*.

[4] R. Garcia, X. Cufi and M. Carreras “Estimating the Motion of an Underwater Robot from a Monocular Image Sequence”, *IEEE Conference on Robots and Systems* (IROS-2001), Hawaii, *to be published*.

[5] R. Garcia, X. Cufí and J. Batlle, "Detection of Matchings in a Sequence of Underwater Images through Texture Analysis", *IEEE International Conference on Image Processing* (ICIP-2001), Thessaloniki, Greece, 2001.

[6] R. Garcia, J. Batlle, X. Cufí and J. Amat, “Positioning An Underwater Vehicle Through Image Mosaicking”, *IEEE International Conference on Robotics and Automation* (ICRA-2001), Seoul, Rep.of Korea, 2001.

[7] X. Cufí and R. Garcia, “Image Stabilisation based on Mosaics”, Workshop on European Scientific and Industrial Collaboration (WESIC-2001), Enschede, The Netherlands, 2001.

[8] R. Garcia, X. Cufí, X. Muñoz, L. Pacheco, J. Batlle, “An Image Mosaicking Method based on the Integration of Grey Level and Textural Features”, *IX Simposium Nacional de Reconocimiento de Formas y Análisis de Imágenes* (SNRFAI-2001), Benicassim, Castellón, 2001.

[9] R. Garcia, X. Cufí, Ll. Pacheco, “Image Mosaicking for Estimating the Motion of an Underwater Vehicle”, *5th IFAC Conference on Manoeuvring and Control of Marine Crafts* (MCMC-2000), Aalborg, Denmark, 2000.

[10] R. Garcia and X. Cufi, “A Review on Mosaicking Systems for Underwater Applications”, Technical Report IIiA 00-16-RR, 2000.

[11] R. Garcia, X. Cufi and J. Batlle, “Motion Detection from a Kalman Filtering Approach”, Technical Report IIiA 99-18-RR, 1999.