Cooperative Assembly for Mobile Manipulators in an Underwater Mission Scenario

Davide Torielli

DIBRIS - Department of Computer Science, Bioengineering, Robotics and System Engineering

University of Genova

In partial fulfillment of the requirements for the degree of

Robotics Engineering

Semptember, 2019

“Choose a job you love,
and you will never have to work a day in your life”
Confucius (maybe)

Acknowledgements

I would like to thank my mother, my father, and my whole family, without whom I would not be here. I would to thank my sweetheart and my love, Sara (aka Cucurucho) which has always supported me and always will do.

I would like to thank the Spanish lab, IRSLab, especially Javier, whom really help me a lot despite every morning at 10:12 he distracted me with Spanish almuerzos. I would also to thank professors from my home country, Professor Giuseppe Casalino, and Ernico Simetti, my two thesis relators. I want also to remember my Spanish Supervisors, Professor Pedro J Sanz and Professor Raul Marin Prades.

I think that I have to thanks also my colleague and friend, Fabio. Also him distracted me a lot with very long telephone calls, anyway they were useful for him to learn something from me =) .

Last but not least, I have to thank myself, because I'm great. Bravo Tori.

Abstract

Contents

List of Figures

Chapter 1

Introduction

Nowadays, different kinds of robot are largely used for underwater missions. A particular type of submarine robot is the Underwater Vehicle Manipulator System (UVMS): an autonomous underwater vehicle (AUV) capable of accomplishing tasks that require a certain level of dexterity, thanks to single or multiple arms.
A really innovative field for underwater missions is the analysis of cooperation between multiple agents, which permits to extend their flexibility. Cooperative robots are two or more robots, identical or different, that coordinate themselves autonomously (i.e. without human intervention) to accomplish various objectives, from mapping an area to assembling an object.

This thesis focuses on a totally unexplored environment: cooperative *peg-in-hole* assembly with underwater manipulators. It is part of the TWINBOT project [TWINBOT (2019)], which is devoted to make a step forward to missions in complex scenarios. Effort is devoted to extend the capability of robots to be able to solve strategic missions. Cooperation is the key in this scenario. Cooperative agents augment the capability of the single, for example to carry an heavy object.

At the time this thesis is developed, the TWINBOT project was in an early stage. So, this works evolves autonomously, always keeping an eye on the main objective of the project: improve capabilities of cooperative underwater intervention robots. So, particular focus will be posed on the cooperation between the two robots. Specifically, this thesis develops a control architecture for the kinematic level. It adopt inverse kinematic algorithms based on task's priority (TPIK), where the peg-in-hole mission is included.
For the detection and pose estimation of the hole, computer vision algorithm are exploited. In particular, stereo-vision methods are implemented, using the cameras of a third robot, who acts exclusively for the vision part.
This pose is only an estimation: to correctly insert the peg into the hole during the final phase, a force-torque sensor is used and information provided by it shared

between the two agents.

Part of this thesis is developed at IRSLab at Universitad Jaume I, Castellòn de la Plana, Spain, during the time I spent as an Erasmus+ 2018/19 student, under the supervision of Professor Pedro J. Sanz and Professor Raúl Marín Prades. The IRSLab is the coordinator of the cited TWINBOT project.
Another part of the work is done at GRAAL at Università degli Studi di Genova, Italy, under the supervision of Professor Giuseppe Casalino and Professor Enrico Simetti.

The architecture is implemented in C++ and the code is avaible on GitHub at the following link: https://github.com/torydebra/AUV-Coop-Assembly.

This thesis is structured as follows. Chapter 1 introduces the problem, recaps the previous work in this field, and states objectives of the work. Chapter 2 defines the theory behind the work. Chapter **??** applies the theory explained to the actual problem, and analyses the results. Chapter 4 focuses on the vision part, explaining methods and discussing results. In Chapter 5, conclusions are given.

1.1 State of the Art

Robots have been massively introduced in various fields to help humans in different tasks. Nowadays, there are strategies to use them in underwater environments. A manipulator (robot arm) is considered to be the most suitable tool for executing sub-sea intervention operations. Hence, unmanned underwater vehicles (UUVs), such as remotely operated vehicles (ROVs) and autonomous underwater vehicles (AUVs), are equipped with one or more underwater manipulators.

During the last 20 years, underwater manipulators have been used for many different sub-sea tasks in various fields, for example, underwater archaeology [Bingham *et al.* (2010); Drap (2012)], marine geology [Wynn *et al.* (2014); Urabe *et al.* (2015)], and military applications [Capocci *et al.* (2017)].

There are many specific tasks where the underwater manipulators are important, from salvage of sunken objects [Cheng Chang *et al.* (2004)] to mine disposal [Fletcher (2000); Djapic *et al.* (2013)]. One particular scenario is towards the oil and gas industry, where underwater manipulators are used for pipe inspection, opening and closing valves, drilling, rope cutting [Christ & Wernli (2013)], and, in general, to reduced field maintenance and development costs [Gilmour *et al.* (2012)]. A recent survey explored the market related to this technology for oil and gas industry [Diaz Ledezma *et al.* (2015)].

1.1.1 Previous Works in Underwater Missions

Since early '90s, research in marine robotics started focusing on the development of underwater vehicle manipulator systems (UVMS). ROVs have been largely used, but they have high operational costs. This is due to the need for expensive support vessels, and highly qualified man power effort. In addition, the pilot which operates the vehicle and the arm experiences heavy fatigue in order to carry out the manipulation task.

For the above reasons, the research started to increase the effort toward augmenting the autonomy level in underwater manipulation. For example, to reduce the operatorâĂŹs fatigue, some autonomous control features were implemented in work class ROVs [Schempf & Yoerger (1992)]. Another solution is to completely replace ROVs with autonomous underwater vehicles (AUVs).

Some pioneering projects in this field carried out in the '90s. The AMADEUS project [Lane *et al.* (1997)] developed grippers for underwater manipulation [Casalino *et al.* (2002)] and studied the problem of dual arm manipulation [Casalino *et al.* (2001)].

The UNION project [Rigaud *et al.* (1998)] was the first to perform a mechatronic assembly of an autonomous UVMS.

Early 2000s showed many field demonstrations. The SWIMMER project [Evans *et al.* (2001)] developed a prototype autonomous vehicle to deploy a ROV mounted on an AUV. This permitted to remove the need of long umbilical cables and continuous support by vessels on sea surface.

This work was followed by the ALIVE project [Evans *et al.* (2003); Marty (2004)], that achieved autonomous docking of Intervention-AUV (I-AUV) into to a sub-sea structure not specifically created for AUV use.

The SAUVIM [Yuh *et al.* (1998); Marani *et al.* (2009)] project carried out the first autonomous floating underwater intervention. It focused on the searching and recovering of an object whose position was roughly known a priori. Here, the AUV weighted 6 tons, and the arm only 65 kg, so the dynamics of the two subsystems were practically decoupled and the two controllers were separated. The mission consisted in the AUV performing station keeping while the arm was recovering the object.

After SAUVIM, a project called RAUVI [Prats *et al.* (2012)] took a step further. Here, the AUV performed a hook-based recovery in a water tank. Again, the control of the vehicle and the arm was separated, even if the Girona 500 light AUV and the small 4-degrees-of-freedom (DOF) arm used had more similar masses than the ones used in SAUVIM.

A milestone was the TRIDENT project [Sanz *et al.* (2012)]. For the first time, the vehicle and the arm were controlled in a coordinated manner [Casalino *et al.* (2012)] to recovery a black-box mockup [Simetti *et al.* (2014a)]. The used task priority solution dealt with both equality and inequality control objectives, although the inequality ones were only-scalar, except for the joint limits. This permitted to perform some manipulation tasks *while* considering also other objectives, for example, keeping the object centred in the camera frame. In this project, only the kinematic control layer (and not also a suitable dynamic one) was implemented.

The PANDORA project [Lane *et al.* (2012)] focused on increasing the autonomy of the robot, by recognizing failures and responding to them. The work combined machine learning techniques [Carrera *et al.* (2014)] and a task priority kinematic control approach [Cieslak *et al.* (2015)]. However it dealt with only equality control objectives, with a specific ad-hoc solution to manage the joint limit inequality task.

The TRIDENT concepts were enhanced within the MARIS project [Casalino *et al.* (2014)]. The used task priority framework [Simetti & Casalino (2016)] permitted to *activate* and *deactivate* equality/inequality control objectives of any dimension (not only scalar ones). This project also extended the problem to cooperative agents [Simetti & Casalino (2017)].

TRIDENT and MARIS concentrated on using the control framework to perform only grasping actions. Recent work [Simetti *et al.* (2018)] analyses how the method can be used in different scenarios, like pipeline inspection and deep sea mining exploration.

The DexROV project [Di Lillo *et al.* (2016)] is studying latencies problems which arise de-localizing on the shore the manned support to ROV operations.

The PROMETEO project [PROMETEO (2016)] plans to improve the use of underwater robotics in archaeological sites. It investigates the manipulation capacity when occlusions of objects can occur and with a wireless communication system to use the robot without umbilical cable.

The ROBUST project [ROBUST (2016)] aims to explore and to map deep water mining sites, through the fusion of two technologies: laser-based in-situ element-analysing, and AUV techniques for sea bed 3D mapping.

1.1.2 The Control Framework

In the '90s, industrial robotics researches focused on how to specify control objectives of a robotic system. This was done especially for redundant systems, i.e. systems with more degree of freedoms (DOFs) than necessary. This surplus is useful to perform multiple, parallel tasks; for example, avoiding an obstacle with the whole arm while the end-effector is reaching a goal. Given that such systems need to complete different goals, it has become important to have a simple and effective way to specify the control objectives.

The task-based control [Nakamura & Hanafusa (1986)], also known as operational space control [Khatib (1987)], defined the control objectives in a coordinate system that is directly linked to the tasks that need to be performed. This idea was followed by the concept of task priority [Nakamura (1990)]. In this theory, a more important task is executed together with a less important task. To accomplish the whole action, the secondary task is attempted only in the null space of the primary one. This means that the secondary task is executed *only if* it does not go against the accomplishment of the first.

This concept was later generalized to multiple task priority levels [Siciliano & Slotine (1991)]. These works putted the position control of the end-effector as the highest prioritized one, while safety tasks (like joint limits) were only *attempted* at lower priority level.

First studies in control of redundant manipulators [Yoshikawa (1984); Maciejewski & Klein (1985)] managed the free residual DOF in such a way to solve the problem of singularity and obstacle avoidance for an industrial manipulator. Another work [Khatib (1986)] introduced the use of potential functions in industrial manipulators and mobile robots.

A different solution [Chiaverini (1997)] proposed a suboptimal approach. The secondary task was solved as if it was alone, but after it was projected in the null space of the higher priority one. To deal with singularities, a variable damping factor was used [Nakamura & Hanafusa (1986)]. This solution was later enhanced and called null-space-based behavioural control [Antonelli *et al.* (2008)].
The approach does not deal with the problem of algorithmic singularities that can occur due to rank loss caused by the projection matrix. Further works [Marani *et al.* (2003); Flacco *et al.* (2012)] focused on this problem.

Since those times, the task priority framework has been applied to numerous robotic systems, other than redundant industrial manipulators. Some examples includes mobile manipulators [Antonelli & Chiaverini (1998); Antonelli & Chiaverini (2003); Zereik *et al.* (2011)], multiple coordinated manipulators [Padir (2005); Simetti *et al.* (2009)], modular robots [Casalino *et al.* (2009)], and humanoid robots [Sentis & Khatib (2005); Sugiura *et al.* (2007)]. Furthermore, a stability analysis for several prioritized inverse kinematics algorithms can be found in [Antonelli (2009)].

The problem of the classical task priority framework, evident in all the previous mentioned works, is that inequality control objectives (e.g. avoiding joint limits) were never treated as such. In fact, the corresponding tasks were always active, like the equality ones. So, for example, also when the joints are sufficiently far from their limits, the fact that the task is active uselessly adds constraints and "consumes" DOFs. Thus, without a transition, the safety control objectives like joint limits could be only considered as secondary. Otherwise, they would consume DOFs, and mission tasks, like reaching a position with the end-effector, can never be accomplished. This led to an undesired situation where safety tasks have a lower priority with respect to non-safety ones.

The challenge in activating (inserting) or deactivating (deleting) a task is that these transitions would imply a discontinuity in the null space projector, which leads to a discontinuity in the control law [Lee *et al.* (2012)]. Thus, in the last decade, researches focused on integrate safety inequality control objectives in a more efficient way.

A new inversion operator was introduced [Mansard *et al.* (2009b)] for the computation of a smooth inverse with the ability of enabling and disabling tasks in the context of visual servoing. But the work only dealt with the activation and deactivation of the rows of a single multidimensional task (so, not including the concept of different levels of priority). The extension to the case of a hierarchy of tasks with different priorities was provided successively [Mansard *et al.* (2009a)]. However, the algorithm requires the computation of all the combinations of possible active and inactive tasks, which grows exponentially as the number of tasks increases.

Another work [Lee *et al.* (2012)] modified the reference of each task that was being inserted or being removed, in order to comply with the already present ones, and in such a way to smooth out any discontinuity. However, the algorithm requires $m!$ pseudo-inverses with m number of tasks. For this reason, the authors provided approximate solutions, which are suboptimal whenever more than one task is being activated or deactivated.

Another approach [Faverjon & Tournassoud (1987)] directly incorporated the inequality control objectives as inequality constraints in a Quadratic Program (QP). According to this, the idea was generalized to any number of priority levels [Kanoun *et al.* (2011)]. At each priority level, the algorithm solves a QP problem, finding the optimal solution (in a least-squares sense). Slack variables are used to incorporate inequality constraints in the minimization process. If the solution contains a slack variable different from zero, it will mean that the corresponding inequality constraint is not satisfied. Otherwise, the inequality constraints are propagated to the next level and transformed into an equality ones (to prevent lower priority tasks from changing the best least-square trade-off found). A similar process is done for the equality constraints. A drawbacks of this approach is that the cascade of QP problems can grow in dimension. Another issue is that the activation and deactivation of tasks are not considered. This last point is important when temporal sequences of tasks are used, for example when assembling objects [Nenchev &

Sotirov (1994); Baerlocher & Boulic (2004)].

Instead of a cascade of QP problems, another research [Escande *et al.* (2014)] proposed to solve a single problem finding the active set of all the constraints at the same time. Due to its iterative nature, the authors proposed to limit the number of iterations to achieve a boundary on the computation time, to be more suitable for a real-time implementation. But this solution is not optimal, and, again, activation/deactivation of equality tasks is not considered.

Improvements are made in the already cited TRIDENT project [Sanz *et al.* (2012); Simetti *et al.* (2014a)], where field trials proved how to consider activation and deactivation of scalar tasks. But the solution still lacks the ability to deal with activation/deactivation of multidimensional tasks, i.e. multiple scalar tasks at the same priority level.

The goal reached by TRIDENT are improved in the MARIS project [Casalino *et al.* (2014); Simetti *et al.* (2014b)], where, among the others accomplishment, task transitions were successfully implemented in the framework. In particular [Simetti & Casalino (2016)], possible discontinuities that can arise are eliminated by a task-oriented regularization and a singular value oriented regularization. Plus, the original simplicity of the task priority framework is retained thanks to pseudo-inverses.

1.1.3 The Peg-in-Hole Assembly Problem

The peg-in-hole is an essential task in assembly processes in various fields, such as manufacturing lines.

This task can be performed following the classical position control method. But this is possible only if precise position of the hole is provided, and the position control error of the robot is zero. In practice, these conditions can only be obtained in specialized scenario. In the case of more versatile robots, such as industrial or underwater manipulators, imprecisions and errors are unavoidable.

To deal with these problems, classical works exploit two kind of instruments: cameras and sensors. With camera(s), the robot can roughly recognize the objective (i.e. the hole) and inspect the overall process. Past researches [Miura & Ikeuchi (1998); Pauli *et al.* (2001)] use this idea to extract boundaries of the object. Another one [Chang *et al.* (2011)] uses visual feedback for a micro-peg-in-hole task (hole of $100\mu m$).

Other approaches perform precise assembly of the parts thanks to force/torque sensors installed on the wrist. A study [Shirinzadeh *et al.* (2011)] successfully accomplishes the assembly detecting the force of contact to compensate the positional uncertainty. Newman *et al.* study [Newman *et al.* (2001)] shows how sensors can be used to build map of force and torque values of each contact point. In another works [Dietrich *et al.* (2010); Abdullah *et al.* (2015)], the location of the hole was estimated using the measured reaction moment occurred by the contact. Another good aspect of the sensors [Oh & Oh (2015); Song *et al.* (2016)] is that they can guarantee stable contact through real-time contact force feedback.

Other proposals [Chhatpar & Branicky (2001); Lee & Park (2014)] try to estimate the state of the contact using joint position sensor. This permits to not use the force/torque sensors on the wrist, which would need highly control frequency, and would increase overall cost and operation time. Some researchers [Park *et al.* (2013)] show that assembly task can be accomplished without contact force information and with inaccurate vision data. The proposed strategy mimics the human behaviour: the peg was rubbed in a point close to the object until the relevant objects mated using compliant characteristics. The compliance allows the robot to softly adapt to the hole [Lozano-PÃľrez *et al.* (1984); Xu (2015)]. A Similar, unexpensive, approach is tested experimentally [Park *et al.* (2017)], without the use of force/torque sensors (i.e. no force feedback), nor Remote-center-compliance devices, and with inaccurate hole information.

1.2 Motivation and Rationale

Sea plays an important role in our societies. Many examples are given in the previous section 1.1. When such kind of environment are so important, robotic usage and exploitation is necessary at different level.
This thesis to improve the current state of art in autonomy of underwater vehicles. Effort in this direction bring the robots to be capable of doing always more complicated task, substituting gradually the remotely operated version of them. The peg-in-hole is an example of these complicated task. In general, robotic assembly problems have been addressed and explored widely, but, to the best of this author's knowledge, no works have been done for cooperatively assembly in underwater scenarios, except for the TWINBOT project [TWINBOT (2019)], which this thesis is part of. Productions related to this problem can help to improve this actual lack and make the technologies to advance.

The aim of this thesis is to developed a kinematical control framework suitable for the discussed scenario. Task Priority Inverse Kinematic is exploited for the control of the single agent, for the cooperation part, and to exploit the information given by vision and force-torque sensor.

1.3 Outline?

The proposed architecture is tested in a simulated environment. The simulator used is UWSim

Chapter 2

Control Framework

Introduction

In this section, the control framework implemented is discussed. The architecture is constituted by two parts:

- The Mission Manager, which job is to supervision the execution of the overall mission. It provides the *action*, a list of control objective that the Kinematic Control Layer must satisfy.

- The Kinematic Control Layer (KCL) focus on provide the system velocities (i.e. vehicle and joint velocities), given the list of control objectives from the Mission Manager.

This architecture is build from the ones used in Simetti & Casalino (2017), Wanderlingh (2018), Simetti *et al.* (2018). The sections 2.1, 2.2, 2.3, 2.4, and 2.5 derive from these works and are here recalled.

2.1 Definitions

In this section, principal used notations are described.

- The system configuration vector of the robot $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{c} \triangleq \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{\eta} \end{bmatrix}$, where $\boldsymbol{q} \in \mathbb{R}^l$ is the l-DOF arm configuration vector and $\boldsymbol{\eta} \in \mathbb{R}^6$ is the vehicle *generalized coordinate position vector*. The first three components of $\boldsymbol{\eta}$ are the position vector $\boldsymbol{\eta}_1 \triangleq \begin{bmatrix} x \\ y \\ z \end{bmatrix}$, with components in the inertial frame $\langle w \rangle$. The last three components of η are the orientation vector $\boldsymbol{\eta}_2 \triangleq \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}$ expressed in terms of the three angles roll, pitch, yaw (applied in the yaw-pitch-roll sequence Perez & Fossen (2007)). The singularity given by this Euler sequence that arise when $\theta = \pi/2$ is handled by a specific control objective (i.e. *horizontal attitude*). (TODO) From the explained definition, it results that $n = l + 6$

- The system velocity vector or the robot $\dot{\boldsymbol{y}} \in \mathbb{R}^n$, $\dot{\boldsymbol{y}} \triangleq \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{v} \end{bmatrix}$, where $\dot{\boldsymbol{q}} \in \mathbb{R}^l$ are the arm joint velocities and $\boldsymbol{v} \in \mathbb{R}^6$ is the vehicle velocity vector. The first three component of $\boldsymbol{v}$ are the linear velocities $\boldsymbol{v}_1 \triangleq \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$ and the last three are the angular velocities $\boldsymbol{v}_2 \triangleq \begin{bmatrix} p \\ q \\ r \end{bmatrix}$, both with components in the vehicle frame $\langle v \rangle$. The vehicle is considered fully actuated, so the vector $\dot{\boldsymbol{y}}$ coincides with the control vector used by the kinematic layer.

2.2 Control Objectives

Let us consider what the robot need to achieve. An *objective* is a job that the robot must accomplish during the mission. Different objectives can be requested at the same time, for example we want the joints to stay away from their physical limits, the robot to maintain an horizontal attitude, and the end-effector to reach a desired pose.

2.2.1 Control Objectives classification

Control objectives can be classified in order of importance in the execution of the mission. Some of them can be more important of others. For example, usually we prefer that the robot do not hurt humans over the reaching of a goal. Another example could be that the robot should first act to not damage itself while accomplish the mission. In general this give the idea that the more important objectives have to been satisfied first, and then, *if possible*, also the less important ones. A general classification based on the *priority* is given here (from the more important class to the less important one):

- *Physical Constraints* objectives. This include interaction with environment (e.g not push against a rigid surface, impose a cooperative tool velocity).
- *System Safety* objectives, e.g. avoiding joint limits or obstacles.
- *Prerequisite* objectives. This is for objectives needed to accomplish the mission, like focus the camera on the object to be grasped
- *Mission* objectives, the actual objective that define the mission, like reach an end-effector position.
- *Optimization* objectives, to choose, among the possible solution (if multiple ones exist) the best one. To example, to choose the one which has the best energy efficiency.

2.2.2 Equality and Inequality Objectives

We consider a variable $\boldsymbol{x}(\boldsymbol{c}) \in \mathbb{R}^m$, dependent on the robot configuration vector $\boldsymbol{c}$, with p the control objective *dimension*. The control objective can be of two types:

- *Equality control objective*, the requirement, for $t \to \infty$, that $\boldsymbol{x}(\boldsymbol{c}) = \boldsymbol{x}_0$.
- *Inequality control objective*, the requirement, for $t \to \infty$, that $\boldsymbol{x}(\boldsymbol{c}) < \boldsymbol{x}_{max}$ or $\boldsymbol{x}(\boldsymbol{c}) > \boldsymbol{x}_{min}$ or $\boldsymbol{x}_{min} < \boldsymbol{x}(\boldsymbol{c}) < \boldsymbol{x}_{max}$.

Please note that here symbols $=, <, >$ refers to element-by-element comparison of the vectors.

2.2.3 Reactive and non Reactive Control Task

For each control objective, there is always an associated *feedback reference rate*. The aim is to drive the variable $\boldsymbol{x}(\boldsymbol{c})$ toward a point $\boldsymbol{x}^*$ where the control objective requisite is satisfied. The used example of *feedback reference rate* is:

$$\dot{\bar{\boldsymbol{x}}}(\boldsymbol{x}) \triangleq \gamma(\boldsymbol{x}^* - \boldsymbol{x}), \quad \gamma > 0 \tag{2.1}$$

That is a simple proportional law, where γ is a positive gain proportional to the desired convergence rate for the considered variable.

To link the considered variable $\boldsymbol{x}(\boldsymbol{c})$ to the system velocity vector, the following relationship is used:

$$\dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\boldsymbol{y}} \tag{2.2}$$

that express how system velocity vector $\dot{\boldsymbol{y}}$ influences the rate of change of the variable. $\boldsymbol{J} \in \mathbb{R}^{m \times n}$ is the so-called *task-induced Jacobian*.
Having the actual $\dot{\boldsymbol{x}}$ as much as possible equal to the desired reference $\bar{\dot{\boldsymbol{x}}}$ is called *reactive control task*.

There are situation where a task has not an associated control objective. For example, it happens when an external command (e.g. an human operator, or an imposed vehicle velocity) provide directly the reference velocities. In this case, there is no desired $\boldsymbol{x}^*$ to reach, and the reference is generated by something else. It this case, we speak about *non-reactive control task*.

2.2.4 Control Objectives Activation and Deactivation

During the execution of a mission, not always each inequality control control objective is relevant. As an example, maintaining joints away from their mechanical limits is a safety task which is needed only when the joints are actually near its limits. When a joint is sufficiently far away, there is no necessity to overconstrain the system imposing a velocity. To deal with this, we speak about *activation* and *deactivation* of control objectives and their relative control task. Let us define the following *activation function*:

$$a(x) \in [0, 1] \tag{2.3}$$

as a continuous, sigmoidal, function, which assumes 0 value outside the validity region of the control objective, and 1 inside it. In between, a smooth transition is present, to gently activate/deactivate the control objective.
For example, considering a scalar ($p = 1$) inequality control objective with the requirement $x(\boldsymbol{c} > x_{min}$ the *activation function* is defined as:

$$a(x) \triangleq \begin{cases} 1, & x(\boldsymbol{c}) < x_{min} \\ s(x), & x_{min} \le x(\boldsymbol{c}) \le x_{min} + \Delta \\ 0, & x(\boldsymbol{c}) > x_{min} + \Delta \end{cases} \tag{2.4}$$

where $s(x)$ is a smooth decreasing function joining the two extreme value 1 and 0, and Δ a value to create a zone where the inequality is satisfied but the activation is between 1 and 0 to prevent chattering problems. Similar definition can be done for the other two kind of inequality control objectives.

In general, when multidimensional control objectives ($m > 1$) are present, the activation takes the form of a diagonal matrix:

$$\boldsymbol{A} \triangleq \begin{bmatrix} a_1 & & \\ & \ddots & \\ & & a_m \end{bmatrix} \tag{2.5}$$

Obviously, for equality control tasks the activation is not defined, because they are always “active”.

For *non-reactive* control tasks, the activation is simply $\boldsymbol{A} \equiv \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}$, being absent the variable $x(\boldsymbol{c})$

2.3 Task Priority Inverse Kinematics

We describe an *Action* $\mathscr{A}$ as list of prioritized control objectives. Each objectives is positioned at a defined priority level k. With this notation, the following symbols are defined:

- $\dot{\bar{\boldsymbol{x}}}_k \triangleq \begin{bmatrix} \dot{\bar{x}}_{1,k} & \cdots & \dot{\bar{x}}_{k_m,k} \end{bmatrix}^T$ is the vector of the reference velocities for the control task k, of dimension k_m.
- $\dot{\boldsymbol{x}}_k \triangleq \begin{bmatrix} \dot{x}_{1,k} & \cdots & \dot{x}_{k_m,k} \end{bmatrix}^T$ is the current rate of change of the k task.
- $\boldsymbol{J}_k$ is the Jacobian relationship which relates the current rate-of-change $\dot{\boldsymbol{x}}_k$ with the system velocity vector $\dot{\boldsymbol{y}}$ as in equation (2.2).
- $\boldsymbol{A}_k \triangleq \mathrm{diag}(a_{1,k}, \cdots, a_{k_m,k})$ is the diagonal matrix of all the activation functions described in section 2.2.4

It is important to notice that different objectives can have same priorities k. In this case, it is possible to simply stack the vectors and matrices to obtain a objective and a related task k that includes both objectives. Without loss of generality, different objectives will be considered always at different priority levels.

The aim of the kinematic layer is to find the system velocity vector $\bar{\boldsymbol{y}}$ that satisfies *as much as possible* the requirements of each objective of the action $\mathscr{A}$. Given the presence of different objectives with different priorities, it must be taken into account to satisfy higher priority objective first. To do this, a sequence of nested minimization problems must be solved:

$$S_k \triangleq \left\{ \arg \mathrm{R}\text{-}\min_{\dot{\bar{y}} \in S_{k-1}} \left\| \boldsymbol{A}_k (\dot{\bar{\boldsymbol{x}}}_k - \boldsymbol{J}_k \dot{\bar{\boldsymbol{y}}}) \right\|^2 \right\}, \; k = 1, 2, \ldots, N, \tag{2.6}$$

where $S_0 \triangleq \mathbb{R}^n$, S_{k-1} is the manifold of solutions of all the previous tasks in the hierarchy, and N is the total number of priority levels. This is the so called *Task Priority Inverse Kinematic* (TPIK). The notation R-min is introduced in Simetti & Casalino (2016). The so called *iCAT* (inequality Constraints And Task transitions) framework solve the (2.6) with the algorithm 1.

Algorithm 1 iCAT

$\boldsymbol{\rho}_0 = \mathbf{0}$

$\boldsymbol{Q}_0 = \boldsymbol{I}$

for k=1 **to** N **do**

$\quad \boldsymbol{W}_k = \boldsymbol{J}_k \boldsymbol{Q}_{k-1} (\boldsymbol{J}_k \boldsymbol{Q}_{k-1})^{\#, \boldsymbol{A}_k, \boldsymbol{Q}_{k-1}}$

$\quad \boldsymbol{Q}_k = \boldsymbol{Q}_{k-1} (\boldsymbol{I} - (\boldsymbol{J}_k \boldsymbol{Q}_{k-1})^{\#, \boldsymbol{A}_k, \boldsymbol{I}} \boldsymbol{J}_k \boldsymbol{Q}_{k-1})$

$\quad \boldsymbol{\rho}_k = \boldsymbol{\rho}_{k-1} + \mathrm{Sat}\left(\boldsymbol{Q}_{k-1} (\boldsymbol{J}_k \boldsymbol{Q}_{k-1})^{\#, \boldsymbol{A}_k, \boldsymbol{I}} \boldsymbol{W}_k \left(\dot{\boldsymbol{x}}_k - \boldsymbol{J}_k \boldsymbol{\rho}_{k-1}\right)\right)$

end for

$\dot{\boldsymbol{y}} = \boldsymbol{\rho}_N$

The special pseudo inverse operator $(\cdot)^{\#, \boldsymbol{A}, \boldsymbol{Q}}$ [Simetti & Casalino (2016)] manages some invariance problems of (2.6); the function Sat$(\cdot)$ [Antonelli *et al.* (2009)] controls the variable saturations. Details of the procedure can be found again in [Simetti & Casalino (2016)].

2.3.1 Notes on Conflicting Objectives

From section 2.2.1, it should be understood that lower priority task are not always satisfied. The problem with this arise when the main mission objective, that is not at the higher priority, can't be never accomplished, thus failing the general mission. This can be the case with an obstacle: the robot may stuck in a point of *local minima* (that is anyway better than crash into it). This is a general problem of all reactive controlling method. The solution must be found at the mission manager level, which should plan another path or another sequence of Actions. This problem is not considered in this work.

2.4 Arm-Vehicle Coordination Scheme

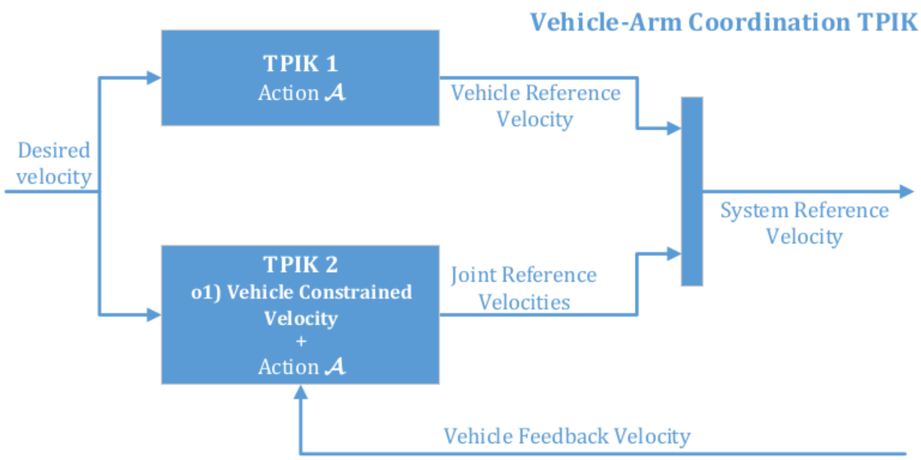


Figure 2.1: A scheme showing the two Task Priority Inverse Kinematics blocks for the arm-vehicle coordination implementation

Inaccuracy in velocity tracking for vehicle can have effects on the arm. A relevant problem arises when disturbances of the floating base, caused by thrusters or its large inertia, propagate and affect the end effector motions [Simetti & Casalino (2017)]. To solve this, a kinematic decoupling of arm and base is done, implementing it within the task priority approach The idea is to have two parallel TPIK as shown in 2.1:

- The first TPIK 1, given the Action $\mathscr{A}$ consider the vehicle together with the arm as a whole full controllable system. From its output $\dot{\bar{y}}$ only the vehicle reference velocity are taken.

- The second TPIK 2 consider the vehicle as totally non controllable. So, a *non-reactive* task (2.2.3) is used at the top of the priority list to *constrain* the output velocities of the vehicles to the actual one. From the total output $\dot{\bar{y}}$, only the manipulator part is taken.
  In this way, the manipulator velocity are *optimized* to follow *at best* the objectives of the action $\mathscr{A}$ considering the *measured* vehicle velocity and their influence on the objectives.

In general, a multi-rate control of arm and vehicle is used, which means that velocities for arm and vehicle are given at different frequency. This schema is suitable for such an implementation: the TPIK 2 can run at higher frequency, updating the manipulator commanded velocities more frequently that the vehicle commanded ones.

2.5 Cooperation Scheme

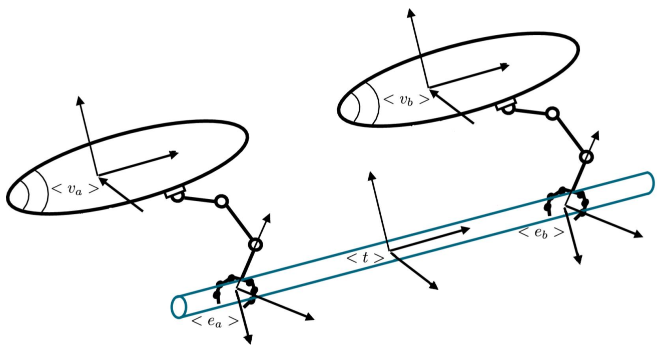


Figure 2.2: The frames of the two cooperative vehicles carrying a common object

The discussion about cooperation is here explained, limiting it to include only two cooperating robotic systems. This is used to transporting a common object. The coordination policy described take care of the bandwidth restriction in underwater scenario. Thus, it deals with the cooperation in a decentralized manner, limiting the exchange of information.
It is assumed that the object is held firmly by both agents, so no sliding happens during the missions. The two robots agree on a shared fixed frame, so, their respective tool frames $\langle t_a \rangle$ and $\langle t_b \rangle$ and the object frame $\langle o \rangle$ are coincident:

$$\langle t \rangle \triangleq \langle t_a \rangle = \langle t_b \rangle = \langle o \rangle$$

In the figure 2.2) the cited frames are shown. The firm grasp assumption imposes that

$$\dot{\boldsymbol{x}}_t = \boldsymbol{J}_{t,a}\dot{\boldsymbol{y}}_a = \boldsymbol{J}_{t,b}\dot{\boldsymbol{y}}_b \tag{2.7}$$

with $\dot{\boldsymbol{x}}_t$ the object velocity with component on $\langle t \rangle$, $\dot{\boldsymbol{y}}_a$ and $\dot{\boldsymbol{y}}_b$ the system velocity vectors of agents a and b as described in section 2.1, and $\boldsymbol{J}_{t,a}$ the system Jacobians of agents a and b with respect to $\langle t \rangle$. These Jacobians tells how the tool velocities $\dot{\boldsymbol{x}}_t$ are affected by system velocities $\dot{\boldsymbol{y}}_a$ and $\dot{\boldsymbol{y}}_b$. Due to the firm grasp assumptions, the tool velocities generates by $\dot{\boldsymbol{y}}_a$ and $\dot{\boldsymbol{y}}_b$ must be equal.

The equation (2.7) derived from the grasp constrain can be expressed in the Cartesian space as:

$$\dot{\boldsymbol{x}}_t = \boldsymbol{J}_{t,a}\boldsymbol{J}^{\#}_{t,a}\dot{\boldsymbol{x}}_t = \boldsymbol{J}_{t,b}\boldsymbol{J}^{\#}_{t,b}\dot{\boldsymbol{x}}_t \tag{2.8}$$

$$(\boldsymbol{J}_{t,a}\boldsymbol{J}^{\#}_{t,a} - \boldsymbol{J}_{t,b}\boldsymbol{J}^{\#}_{t,b})\dot{\boldsymbol{x}}_t \triangleq \boldsymbol{C}\dot{\boldsymbol{x}}_t = \boldsymbol{0} \tag{2.9}$$

$$\dot{\boldsymbol{x}}_t \in ker(\boldsymbol{C}) = Span(\boldsymbol{I} - \boldsymbol{C}^{\#}\boldsymbol{C}) \tag{2.10}$$

The kernel of $\boldsymbol{C}$, called *Cartesian Constraint Matrix*, express the space of achievable object velocities at the current configuration.
The idea of the scheme is to put a non-reactive task at the top of the priority, to constrain the desired object velocity $\dot{\bar{\boldsymbol{x}}}$ in this subspace. In this way both agents can follow this desired object velocity despite the different situation caused by other objectives.

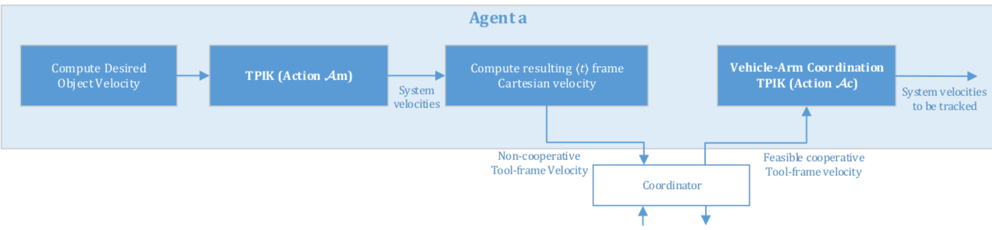


Figure 2.3: The cooperation algorithm with its different steps

The algorithm, schematized in fig. 2.3 proceeds as follows:

- In the first step, the two agents run the algorithm of 2.3 as they were the only to act. So, we have:

$$\dot{\boldsymbol{x}}_{t,a} = \boldsymbol{J}_{t,a}\dot{\boldsymbol{y}}_a, \qquad \dot{\boldsymbol{x}}_{t,b} = \boldsymbol{J}_{t,a}\dot{\boldsymbol{y}}_b \tag{2.11}$$

  where, in general, the two *non cooperative* tool velocities are different: $\dot{\boldsymbol{x}}_{t,a} \neq \dot{\boldsymbol{x}}_{t,b}$.

- The tool velocities are exchanged (i.e. sent to the coordinator) and a *cooperative* tool velocity is computed:

$$\dot{\hat{\boldsymbol{x}}}_t = \frac{1}{\mu_a + \mu_b}(\mu_a\dot{\boldsymbol{x}}_{t,a} + \mu_b\dot{\boldsymbol{x}}_{t,b}), \qquad \mu_a, \mu_b > 0 \tag{2.12}$$

$$\mu_a = \mu_0 + \|\dot{\bar{x}}_t - \dot{x}_{t,a}\| \triangleq \mu_0 + \|e_a\|,$$
$$\mu_b = \mu_0 + \|\dot{\bar{x}}_t - \dot{x}_{t,b}\| \triangleq \mu_0 + \|e_b\|, \tag{2.13}$$
$$\mu_0 > 0$$

where $\dot{\bar{x}}_t$ is the ideal velocity that, if applied, would asymptotically take the tool to the desired goal.
The *cooperative* tool velocity is a *weighted* compromise between the two *non cooperative* ones. The *weights* μ_a, μ_b given more freedom to the robot which meet the highest error e. This error is a way to understand how much one robot is in difficult in tracking the *ideal* tool velocity $\dot{\bar{x}}_t$.

- The new *cooperative* tool velocity $\dot{\hat{x}}_t$ is not, in general, a *feasible* velocity that both vehicle can perform. So, an additional passage is required:

$$\dot{\tilde{x}}_t \triangleq \left(I - C^{\#}C\right)\dot{\hat{x}}_t \tag{2.14}$$

with C defined in (2.9).

- Each agent run a new TPIK procedure, identical to the first one, but with a *non-reactive* control objectives to track the *feasible cooperative* velocity $\dot{\tilde{x}}_t$. The output of the two agents algorithm, $\dot{\bar{y}}_a$ and $\dot{\bar{y}}_b$ will be the final velocity which the kinematic layer provide.
Moving the equality control objective to make the end effector reach the goal at the top of the hierarchy does not influence the safety tasks. This property is proven in Wanderlingh (2018).

In this described method, the only information that the agent must exchange are the *non-cooperative* velocities $\dot{x}_{t,a}, \dot{x}_{t,b}$, the feasible ones $\dot{\tilde{x}}$, and the constrain matrix C. Furthermore, even less data can be exchanged if the *coordinator* is a procedure which runs on a robot, and it is not on an external node.

2.6 Force-Torque Considerations

When a robot interacts with the environment, each contact generates forces on them. Mission like an assembly task can't be studied in a properly manner without some considerations about these forces. In a peg-in-hole Mission, contact between the peg and the hole will be transferred through the whole kinematic chain until the floating base, causing disturbance to the whole robotic system.
Let us define $f \in \mathbb{R}^3$ and $m \in \mathbb{R}^3$ as:

$$f = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \qquad m = \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} \tag{2.15}$$

being the $\boldsymbol{f}$ and $\boldsymbol{m}$ the resultant forces and torques (projected on the tool frame $\langle t \rangle$) of all the forces and torques acting on the tool. These disturbances generate joint velocities $\delta\dot{q} \in \mathbb{R}^l$ [Siciliano *et al.* (2009)]:

$$\delta\dot{q} = ({}^{lin}\boldsymbol{J}_t)^T \boldsymbol{f} + ({}^{ang}\boldsymbol{J}_t)^T \boldsymbol{m} \tag{2.16}$$

where lin and ang stand for *linear* and *angular* part of the Jacobian described in 2.3; respectively the top-left 3×3 and the bottom-right 3×3 submatrices.

When the peg is inside the hole, but not aligned perfectly, contact between the external surface of peg and the internal side of the hole may happen, while the robot is inserting it. This can be the case when the estimation of the hole pose is not perfect and the peg is driven not exactly at the centre of the hole. The resultant forces and torques, transferred to the whole arm with equation (2.16), will naturally driven the peg to be aligned correctly. However, if these forces and torques are not considered by the control architecture, it would cause a persevering chattering because the task will persist in the erroneous direction.
A possible solution is to insert a new objective into the TPIK procedure. The aim of this objective is to zeroing the forces and torques acting on the peg, generating properly joint commands
The *feedback reference rate* for it will be:

$$\begin{bmatrix} \dot{\bar{x}}_f \\ \dot{\bar{x}}_m \end{bmatrix} \triangleq \begin{bmatrix} \gamma_f \\ \gamma_m \end{bmatrix} \begin{bmatrix} 0 - \|\boldsymbol{f}\| \\ 0 - \|\boldsymbol{m}\| \end{bmatrix} \qquad 0 < \gamma_f < 1, \quad 0 < \gamma_m < 1 \tag{2.17}$$

where $\|\boldsymbol{f}\|$ and $\|\boldsymbol{m}\|$ are the norms of the forces and torques vectors $\boldsymbol{f}$ and $\boldsymbol{m}$. Gains smaller that 1 are necessary because big gains would mean too big velocities provided to joints. Being the simulation only kinematic, these velocities are *immediately* followed by the joints without any saturation.
The norms f and m are used instead the full 3-dimensional vectors $\boldsymbol{f}$ and $\boldsymbol{m}$. This is done to not overconstrain the system and let more freedom to lower priority task. Even with norms, the control objective is obviously satisfied.
The *feedback reference rate* is intended to be like additional velocity that the tool must follow. Thus, the Jacobian of this new task will be equal to the $\boldsymbol{J}_t$ introduced in 2.3. The difference is that we have to divide linear and angular part, and pre-

multiplying it for the normal vector of $\boldsymbol{f}$ and $\boldsymbol{m}$:

$$\begin{bmatrix} \boldsymbol{J}_f \\ \\ \boldsymbol{J}_m \end{bmatrix} \triangleq \begin{bmatrix} \dfrac{\boldsymbol{f}}{\|\boldsymbol{f}\|} \; {}^{lin}\boldsymbol{J}_t \\ \dfrac{\boldsymbol{m}}{\|\boldsymbol{m}\|} \; {}^{ang}\boldsymbol{J}_t \end{bmatrix} \tag{2.18}$$

with both $\boldsymbol{J}_f, \boldsymbol{J}_m \in \mathbb{R}^{1\times l}$

Chapter 3

Control Architecture: Methods & Results

In this chapter, experimental set-up is described, and results given and discussed. The architecture is tested on a well defined scenario.
It is made up of two I-AUV's Girona 500 AUV underwater vehicles, each one equipped with a CSIP Robot arm5E (4 DOF arm with a parallel yaw gripper). The final goal is to successfully coordinate the robots in such a way that the peg, hold by both manipulators, is inserted correctly in the other piece, fixed in the environment.

The chosen strategy divides the problem in two phases: Hole Detection and Insertion. In the first preliminary step are done to detect the hole. A third robot, not used for manipulation task, is in charge to exploit vision to estimate the pose of the hole. Detail about this are given in section 4. The second phase explores the problems inherent to the interaction between the peg and the hole, and the communication between the carrying agents. This is described in this chapter.

3.1 Simulators

A bit effort has been spent to choice a suitable simulator for the case. At the end, UWSim [Prats *et al.* (2012)] was chosen. It is a simulator largely used for this kind of scenarios, which visualize a virtual underwater scene. It provides a different variety of useful sensors (e.g. the used force-torque sensor and the cameras), but also others can be added. It is fully integrate in ROS, which made it really easy to use. ROS is used as simulator interface: through ROS messages, we can send commands to the robots and we can receive information from the going on test. Contact physics is implemented integrating the physics engine Bullet with OSG through OSGBullet. To further details about how the simulation is implemented, especially the contact physics part, please refers to the documentation of the cited

software. The cons in using UWSim is that the simulations is fully kinematic, so no dynamics interaction ar present. This means, for example, that velocity sent to the robot are immediately accomplished, and that we can't simulate the real physics while grasping a real object. For the scope of this thesis, this lack is not important because dynamics is not considered. Furthermore, the only needed dynamic part, i.e. how the contact between the tool and the hole affect the whole manipulator chain, can be simulated at kinematic level thanks to the information provided by the force-torque sensor, as explained in section 2.6.
To fill the lack of dynamic of UWSim, a good alternative can be FreeFloatingGazebo [Kermorgant (2014)]. In truth, this simulator is a plug-in for Gazebo and UWSim; it integrates them in order to achieve both dynamic (thanks to Gazebo) and visually realistic I-AUV simulation (thanks to UWSim). The interface used to communicate with the simulation is the same of UWSim, so ROS and its messages, which make it easy as UWSim to use.
This plug-in has been taken into consideration for dynamic test, that are not evaluated due to the lack of time, but can be certainly used for further works. Gazebo is a generic simulator widely used in all robotics fields. It is the de-facto simulator for ROS. Seen its purpose, it is not a ready-to use simulator for underwater environment, and can be only a starting point to build a software to simulate this particular scenario (as it is done by FreeFloatingGazebo).
Also other simulators, V-REP [E. Rohmer (2013)] and Webots [Michel (2004)] have been taken into consideration but discarded for same "not ready-to-use" reason like Gazebo.
An interesting simulator is USV simulator [Paravisi *et al.* (2019)] which takes the best from UWSim, Gazebo and FreeFloatingGazebo to implement realistic simulation. This is a really recent and in development project, and however it is focused more on surface vessels dynamics.

More details and comparisons are available in Cook *et al.* (2014) and Paravisi *et al.* (2019), and a schematic recap taken from Paravisi *et al.* (2019) is visible in fig. 3.1.

Simulator	Waves	Buoyancy	Water Currents	Wind Currents	Thruster Underwater	Thruster above Water	Foil
UWSim	$\surd$	$\surd$	$\times$	$\times$	$\surd$	$\times$	$\times$
Gazebo	$\times$	$\times$	$\times$	$\times$	$\surd\surd$	$\surd\surd$	$\times$
Freefloating Gazebo	$\surd$	$\surd$	$\surd$	$\times$	$\surd\surd$	$\surd\surd$	$\times$
VREP	$\surd$	$\surd$	$\times$	$\times$	$\surd$	$\surd\surd$	$\times$
RobotX Simulator	$\surd$	$\surd\surd$	$\times$	$\surd$	$\surd\surd$	$\surd\surd$	$\times$
USVSim	$\surd\surd$	$\surd\surd$	$\surd\surd$	$\surd\surd$	$\surd\surd$	$\surd\surd$	$\surd\surd$

Figure 3.1: Schematic recap of simulation comparison taken from Paravisi et al. (2019). $\times$ stands for no implemented feature; $\surd$ for a feature that is a discrete representation of the real one; $\surd\surd$ for a good representation of the real one. More details on how each feature is evaluated are available in the original paper.

3.2 Assumptions

It is important to detail the main assumptions (related to the control architecture) made. A lot of problems, that it is necessary to take into account in a real environment, are not explored. This is necessary due to the difficulty of the particular scenario chosen.

- Simulation is only kinematic. This implies, for example, that the commanded velocity to the vehicle and the arm are accomplished *instantaneously* and *perfectly*. Another implication is that the movements of arm and of the vehicle don't influence each other at all. The only "dynamic" implemented is caused by collision between the peg and the hole, which transfer the forces and torques acting on the peg along the whole arm. Disturbances of this type on the vehicle are neglected. (TODO

- The initial configuration is with the peg already *correctly* grasped by both robots. Also, the position of the grasped point and the peg dimension are known: this imply that relative position between robot and tip of the peg is *perfectly* known. This initial condition is chosen because the grasping phase and problems arising during cooperative transportation have been explored in others cited project like MARIS and ROBUST (e.g in the work Simetti & Casalino (2017)) and also as part of the on-going project TWINBOT.

- The peg is firmly grasped by both robot, no slippery is take into account. In truth, here an important detail must be explained. To the work purpose, and for simulator limitations, two pegs are modelled, each one like a fixed part of each robot. So it is truth that each peg is fixed respect to each agent, but the two pegs (perfectly overlapped at the beginning) can distance themselves. This is a useful benchmark to understand how good is the performance: the more the pegs distance themselves, the more we would apply a stress on the object (and on the arms) in a real situation.

- Pose (linear and angular displacement) of the two carrying robots and of the vision robot respect a common inertial frame is known. In real situation, knowing the position underwater is really an issue and it is never really precise. This information can be provided, for example, thanks to some mappings of the seafloor, to find a common interest point which refer to. Note that it is not important know the pose of the robot respect to a point above the water surface (that can be done thanks to information shared with surface vessels, for example as explored the WiMUST project [Abreu *et al.* (2016)]). The important thing is to know relative pose of the robots to a common node, that can also be underwater. This is needed to make the vision robot share correctly the estimated pose of the hole.

- No real communication issues between the two carrying robot are taken into account. The presence of water put important issues in real situation; for example with water *full-duplex* communication (i.e. sharing data *at the same time*) is impossible, and in general a there is lower bandwidth than in the air. Some experiment in simulated environment with different methods of underwater communication are detailed in Simetti & Casalino (2017).
  The issues about communication are taken indirectly using a cooperative scheme (explained in section 2.5) which permits to exchange few information between the two carrying agents.

Others assumptions, more related to the vision part, are detailed in section 4.1.

3.3 Tools

The Control Architecture is implemented in C++ language, using various libraries. In this section, the most relevant libraries and tools used for the software are described. Please note that a particular section (section 3.1) is dedicated to the chosen simulator UWSim.

- **ROS** (Robot Operating System), the well-known robotic middleware. It is used to communicate with the simulator, so to send commands to robots and receive information by the on-going simulations (e.g. robots states, information from sensors, streaming images from cameras).

- **Eigen** [Guennebaud *et al.* (2010)], a C++ library for linear algebra. It is very useful to deal with matrices computations and management in any C++ software.
- **CMAT**, another C++ library, implemented at GRAAL. It implements the core functions for the TPIK method, detailed in Simetti & Casalino (2016).
- **Orocos KDL** a package to deal with kinematic and dynamic chain. In the case of this thesis, it is used to compute the Jacobian of the robots for each of their configurations.

3.4 qui spiega il task inseriti e in particular il force task

Chapter 4

Vision: Methods & Results

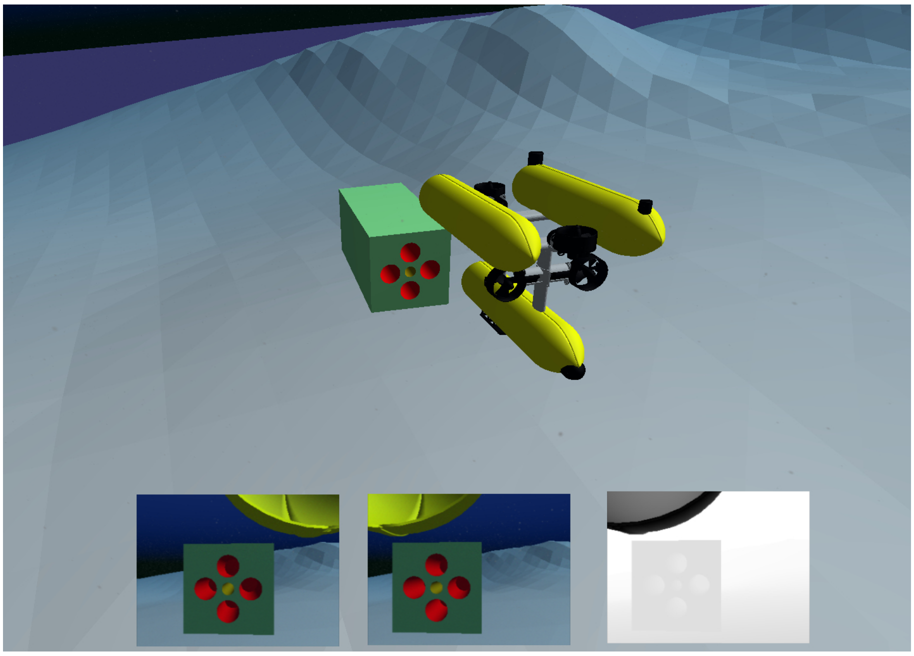


Figure 4.1: The vision Robot watching the hole. The hole is in the centre of the cuboid. The red holes are only present to help vision algorithms. Below, what the right and left camera are seeing. Note that also a depth left image is provided, but the left RGB camera and the depth camera are not used together.

Before the twin robots can approach the hole, its position must be know, at least roughly. In this chapter, the pose estimation of it is discussed.
In the considered scenario, a third robot is present. Its duty is exclusively to *detect* & to *track* the hole. In the simulation, another Girona 500 AUV is used for this job, without the arm. Is evident that, in real scenario, a littler and more efficient

robot should be used for the vision, see that no manipulation task are needed. In fact, in the original TWINBOT [TWINBOT (2019)] simulation, a smaller BlueROV is present, as can be seen in (TODO) However, in this case, another Girona 500 is used to not deal with another robot model.
The *vision* robot is equipped with two identical cameras which point in front of it. They are used as:

- Two distinct cameras, independent one of the other.
- As stereo cameras, thus exploiting stereo vision algorithms.
- As RGB-D camera, i.e., a stereo vision couple where the left one is a RGB camera and the right one a Depth camera.

The job is done into two phases: *Detection* (section 4.3) and *Tracking* (section 4.4).

4.1 Assumptions

For the sake of simplicity, some assumptions are made:

- Known *intrinsic* camera parameters. These parameters are used by algorithms to take into account how the single camera see the scene. The (known) distortion is zero.
- Known *extrinsic* camera parameters, i.e. the position and orientation of cameras (respect the vehicle), and thus the relative pose (the transformation matrix) from one camera to the other (needed for stereo vision algorithm).
- No external disturbances for the images, such as light reflections underwater or bad visibility.
- Hole model known. This means that dimensions of the cuboid which contains the hole are known. Further explanation about this are given successively in section 4.4.
- A "friendly" cuboid structure of the hole. The front face is coloured and additional holes are present, as can be seen in fig. 4.1. This help both the *detection* phase and the *tracking* phase.

About the robot, other assumption are:

- the pose of the vehicle respect the inertial frame is known. (TODO?AS explained?? se si linka sez). This permits to know the estimation of the pose of the hole respect the inertial frame, to directly send the robot which are carrying the peg.

- The initial position of the robot is such that it is facing the front face of the hole. It must be noticed that methods explained in the next sections can be adapted to relax this hypothesis. For example, the robot could turn around z-axis until the hole is detected. (TODO?? Also, good results are obtained when the robot not exactly face directly the cuboid, but, for example, it is on its side, looking at front face and a side one.

- Once the robot has tracked the hole and the pose sent to the twin robots, it must go away to not interfere with the insertion phase. This is done through keyboard (as a ROV) but it is not difficult to improve the code to let him go away autonomously. It must be noticed that, thanks to the *tracking* algorithms, if the robot moves (because it is commanded to do so, or for water currents) the pose estimation is still good.

4.2 Tools

To deal with the pose estimation, some external tools are used. In this section they are listed.

- **OpenCV** (Open Source Computer Vision Library) [Bradski (2000)], an open-source BSD-licensed library that includes several hundreds of computer vision algorithms. It is used mostly for the detection part, even if some of its functionalities are used also by ViSP (for example for keypoint tracking).

- **ViSP** (Visual Servoing Platform) [Marchand *et al.* (2005)], another open source library that allows developing applications using visual tracking and visual servoing techniques. It is helpful because is more specific for robotic fields and easier to use than OpenCV. It is used for the tracking phase.

- **PCL** (Point Cloud Library) [Rusu & Cousins (2011)] a library for 2D/3D image and point cloud processing. In this work, is used by ViSP when depth images are used. However, further works can used as another help to deal with the vision part.

4.3 Detection

Object Detection means detecting a particular shape (i.e. the *object*) in the scene. This is important to initialize the tracking algorithm used. In fact, for the tracking algorithm used successively, the detection part must provide a correspondence between some pixels in the 2D image and some points in the 3D object shape. It is

important to notice that the needed 3D coordinates refer to the object frame, and not to an "external" frame. Seen that the object model is assumed to be know, the 3D coordinates of some point directly derive from this assumption.
Four points is the minimum number of point accepted by the tracking algorithm. The more the point are, the more the tracking is good. Plus, point should lying on different surfaces of the object, to have better results. Anyway, good tracking result are obtained also not considering these two aspects. The four points chosen are the corners of the front face of the cuboid, where there is the hole.
As an example, the .init file with the 3D coordinates is like the one in file 2.

File 2 The .init file describing the position on the 4 corners of the front face, respect to a frame positioned in the centre of the hole, with x-axis going inside the hole, y lying along the surface pointing on the right, z pointing down to the seafloor.

```
4             # Number of points
              #xyz with x going in, y on the right, z down. Measure is meter
0 -0.4 -0.4   # top right corner
0 0.4 -0.4    # top left
0 0.4 0.4     # bottom left
0 -0.4 0.4    # bottom right
```

The work of Detection is to provide 2D coordinates of the image pixels that correspond to the 3D points of file **??**. This must be done for each camera, except for the Depth one (when used).
Two methods are evaluated: *Find Square* (section 4.3.2) and *Template Matching* (section 4.3.3). A third method, in which the 2D coordinates are precise as much as possible (selecting by hand the four pixels containing the corners), is used to have a benchmark for the other two and to analyse the tracking when 2D Coordinates are almost perfect (section 4.3.1).
Details of how each function works and explanation of the computer vision algorithms used are not provided here, to not go outside the scope of the thesis.
Other methods and functions are briefly explained in Appendix A. Another, not explored, method can be to use tags code on the cuboid surface. However, in underwater situations this can be difficult to be put in practice.

4.3.1 Already known Coordinate Method

As explained, with this method the 2D coordinates are perfectly known. This is done by letting the user to click on the 4 pixel which contains the corners. Given that the image is made by discrete pixels, is impossible to have an ideal point which is exactly the corner, but the errors for this are not noticeable.

4.3.2 Find Square Method

This method is taken from an OpenCV tutorial.
A rough explanation of how the method works is presented:

- This method looks in each image channel (unique if is a gray image, three if is a colored image) to find squares.
- First, it pre-processes the image to reduce noise, down and up scaling it.
- Then, *findContours()* is called to retrieves contours of the shape with the algorithm described in Suzuki & Be (1985).
- Each contour is approximated to be more like a regular polygon, with less vertices and edges.
- Finally, the algorithm looks if the shapes are similar to squares/rectangles. This is done checking if the internal angles are almost 90 degrees.
- The returned shapes are described by their four corners, that is what we are looking for. An additional function is called to be sure that the order of the returned corners is the same order of the points described in the .init file, otherwise correspondences are obviously erroneous.

4.3.3 Template Matching Method

Template Matching means to find a pattern (in this case, the face of the hole) inside a scene. So, an additional image of the square face of the hole is needed.
The code developed follow an OpenCV tutorial.
In brief, *template matching* finds the point in the scene which as more similarity (or less dissimilarity) with the provided template. This is done considering intensity values of the pixels in the neighbourhood area of a center pixel. In practice, the template is shifted all over the scene image and some calculations for each new template shifting are done. Various formulas to compute similarity (or dissimilarity) are provided by OpenCV and are detailed here. The choosen one in the experiment is the so called *squared differences* (the first in the link).
It is important to scale up and down the template and to compute various time the similarity. This because usually template size is not equal to the size of the object in the scene. For each scaling, a best similarity point is detected. Then, all the similarity points are compared and the best one are chosen. At the end, a rectangle with the template (scaled) dimensions is build considering the best point as the centre. The corners of the rectangle are the 4 points which we were looking for.

4.3.4 Detection Results

In this case, the find square method give the best results. As can be see in fig. 4.2, differences from the ideal method and this one are barely visible. In the way it works, it should be noticeable that this method gives good results only if the camera approximately face the cuboid structure at the front. (TODO? as can be see se ti metti di lato...). If the side face is more visible, it will be the one detected. So, we must know which side the robot is facing to give the 3D correspondence points in the .init file.
In addition, this method is suitable if no other squares of similar dimensions are present. If so, further work is needed to discriminate them. Also, is not suitable with other kind of shapes (a pipe hole, for example). It is also important to point out that sometimes the method fail to find any shape in the right image. This happened approximately 30% of the time, and could show a very bad robustness and low predictability of the method. However, the fail is obviously detectable and another trial can be done easily.
The template matching is less precise than the previous. Plus, if the face is view from a different angle, other template image is needed, with an orientation similar to what the robot is seeing. In general, lot of template images at different angles are needed. Otherwise, some processing of the template image is necessary to orient it in a different way. Also, building the shape around the centre point is more difficult, if this shape is not a square. Anyway, the template method is more general because can be used also for different shapes (e.g. a hole of a round pipe).

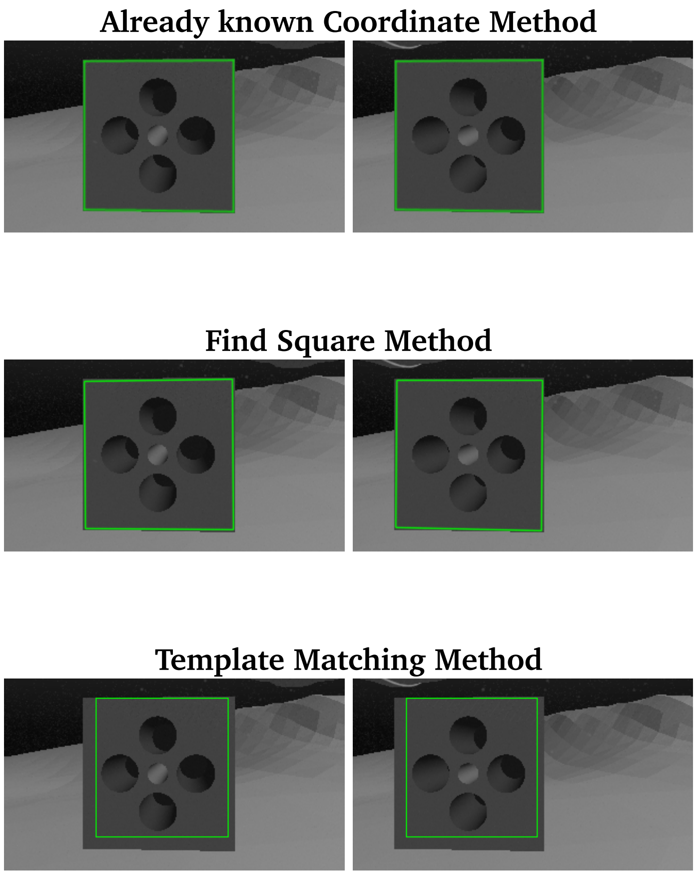


Figure 4.2: Results of the three different detection method. The green rectangle is the estimate position of the square. The output of the detection step are the four corners of the green rectangle.

4.4 Tracking

Object Tracking means to follow the motion of an object of interest during time. Both the object and the camera can be mobile, even if, in this case, only the cameras are moving (actually, the robot moves, the cameras are rigidly attached to its body). Tracking an object is usually done to estimate its pose respect to the camera frame.

In this case, we speak about a *markerless model-based* tracking. Thus, the object model must be provided. In this scenario, it is sufficient to give to the algorithm the 3D dimension of the cuboid structure of the hole, with a circle in the front face. Using ViSP, the format required for the model is *.cao*, which sintax is described here. As explained in section 4.3, the algorithm must know the position of *at least* four point belonging to the cuboid. In the experiments, the provided ones are the 4 corners of the front face.

Three different trackers have been tried: *Two Mono Cameras Tracker* (section 4.4.1), *Stereo Camera Tracker* (section 4.4.2), and *Stereo Depth Camera Tracker* (section 4.4.3).
A tracker is linked to each camera. For RGB cameras, it can be of three types: *edge-based* [Comport *et al.* (2006)], *keypoint-based* [Pressigout & Marchand (2007)] or a mix of both. During the experiment, the hybrid method emerged as the most precise, so all the results in section 4.4.4 refer to this one. For the depth camera used in *Stereo Depth Camera Tracking*, the tracker type can be *normal* or *dense* [Trinh *et al.* (2018)]. Being the *dense* one more robust, it is the only one to has been considered. Please note that it is also computationally heavier for large matrix computations, but speed performance are not considered here.

4.4.1 Two Mono Cameras Tracking

This method derived from the ViSP *Markerless generic model-based tracking using a color camera* tutorial.
The implementation is straightforward: after setting the trackers (i.e. giving edge and keypoint detection parameters, camera parameters, and 2D-3D correspondence of the four corners), at each loop the tracker estimates the transformation matrix between each camera and the object.
In this method, the left and right cameras are independent. Thus, each one provides a different pose estimation. It is not so easy understand when one camera provides better results that the other. Anyway, it should be easy to understand when one camera fails completely in tracking the object.

4.4.2 Stereo Camera Tracking

This method derives from the ViSP *Markerless generic model-based tracking using a stereo camera* tutorial.
The code is analogous to the previous one, except that in this case also the relative pose between each camera must be provided. If this is unknown, some method for stereo calibration must be used.

4.4.3 Stereo Depth Camera Tracking

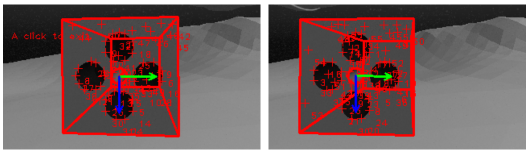
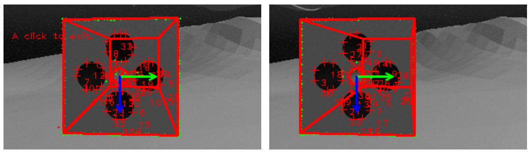
This method derived from the ViSP *Markerless generic model-based tracking using a RGB-D camera* tutorial.
This method is similar to the previous one, except that the right camera is now a depth one, thus providing depth images. The functions used for depth images need

the support of another library, PCL (Point Cloud Library) [Rusu & Cousins (2011)]. Another exception is that the depth camera does not need to initialize the 2D-3D correspondences, so the *Detection* step has to be done only for the left camera.

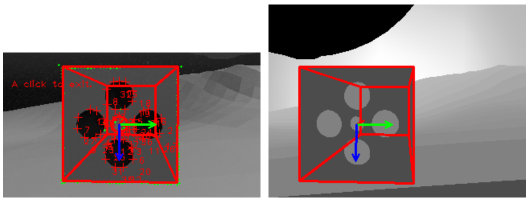
4.4.4 Tracking Results

In this section, performance of the three tracker are evaluated. For each one, the three different types of detection initialization (explained in 4.3) are considered to see the effect of detection error on each tracker.
Experiments have been conducted with lot of simplifications: no disturbance, no cameras distortions, very good visibility, nice object shape. Results described here can give only an idea on how to proceed in a more realistic scenario.
In the scenario, the robot is perfectly still while tracking the object, and it is in front of the object, slightly on the right. The original images taken from cameras are cut to delete a region where part of the vehicle is visible. This is done to make this part not interfere with the algorithm. In the depth image, this is not necessary, because there is no interference.
In fig. 4.3 the detected shape and the estimated frame are drawn on the camera images. Differences are barely visible in the first two initialization methods. With the template matching method, bad initialization (shown in 4.3.4) is paid in tracking result, especially in the depth case. This is clear in fig. 4.6. With this initialization, the depth-stereo method is even worse than the monocular case. This can be due to the fact that the depth image is not initialized with 2D-3D correspondence; thus paying more the initialization error being done only in the left image. So, it is showed that it is not always better to have a RGB-D camera instead a normal RGB. This is an interesting result and should be further explored with more realistic scenarios.
With good initialization, (fig. 4.4, fig. 4.5), the stereo methods have similar results, overall better than the mono case. which however has not bad performance. Another interesting result is that, in the monocular case, the position of the camera influence the results. This is because different view angle obviously provides different tracking performance.
In the plot of the errors (fig. 4.4, fig. 4.5, fig. 4.6), lot of variations during time can be seen, although the robot is still. This is due to the nature of the tracking algorithm, which continuously estimates the pose for each image received by the camera. However, it must be noticed that the variations are little for both linear and angular parts.

Already known Coordinate Initialization

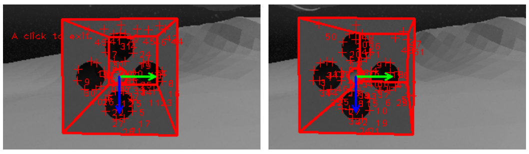
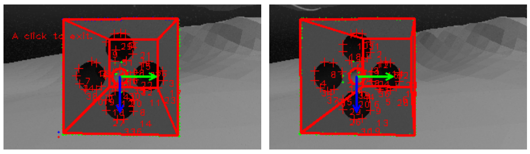



Mono Cameras Case *Stereo Cameras Case*

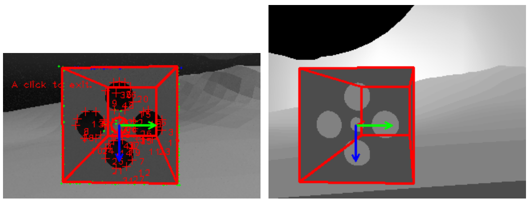


Stereo Depth Camera Case

Find Square Initialization

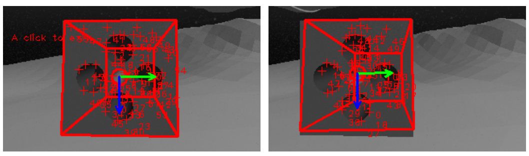
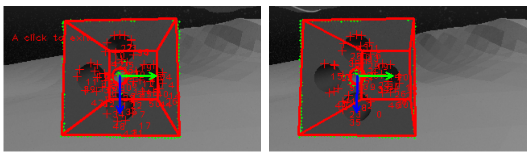



Mono Cameras Case *Stereo Cameras Case*

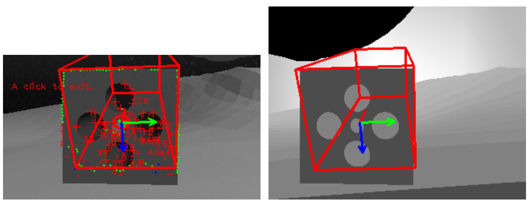


Stereo Depth Camera Case

Template Matching Initialization

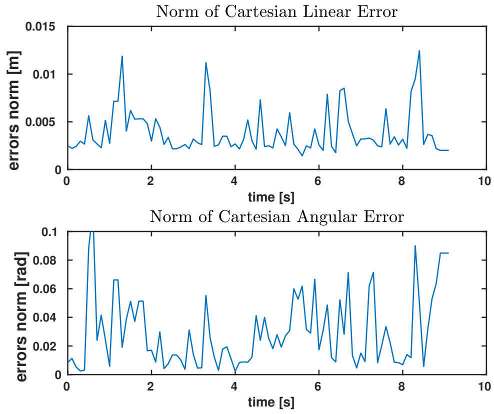
Mono Cameras Case *Stereo Cameras Case*



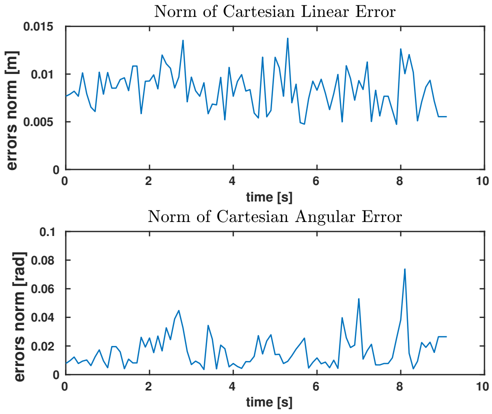
Stereo Depth Camera Case

Figure 4.3: Tracking Results. Red lines are the contours of the model where is estimated to be ; red cross are the point tracked by the algorithm. The arrows represent the estimated object frame: green for y-axis, blue for z-axis, red for x-axis (which go inside the hole and is barely visible). Green dots are the tracked correspondent points between left and right images, thus they are present only in the stereo cases.

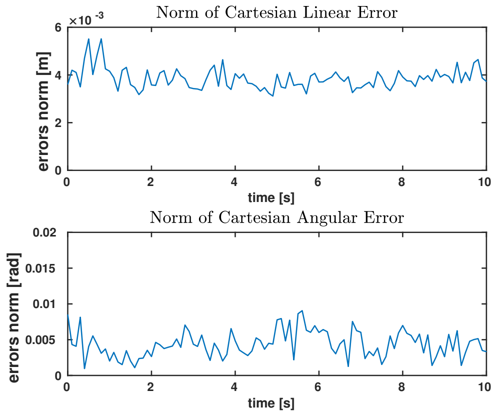
Already known Coordinate Initialization



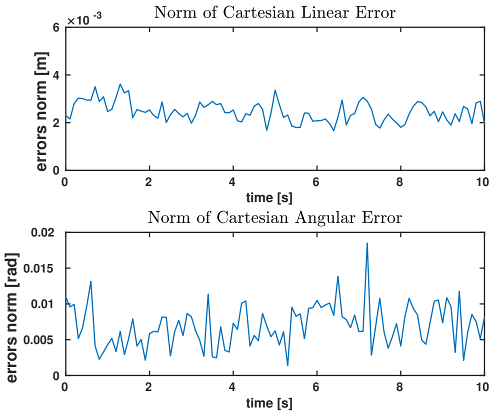
Mono (left Camera) Case



Mono (right Camera) Case



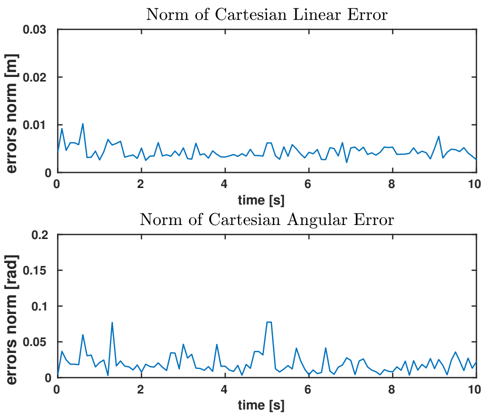
Stereo Camera Case



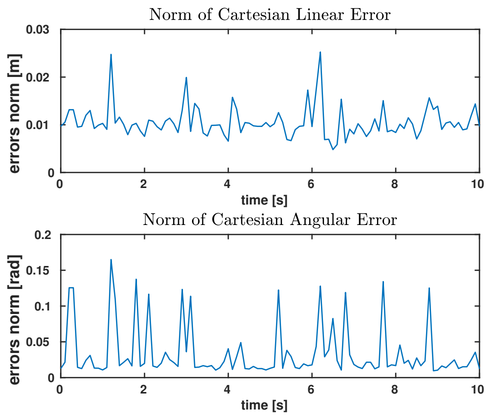
Stereo Depth Camera Case

Figure 4.4: Linear and Angular Error (in norm) between true pose and estimated pose, with the best initialization of detection step possible.

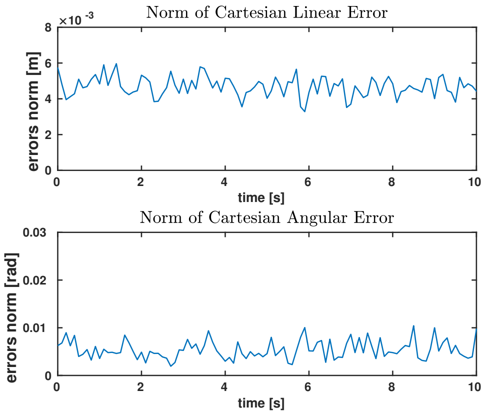
Find Square Initialization



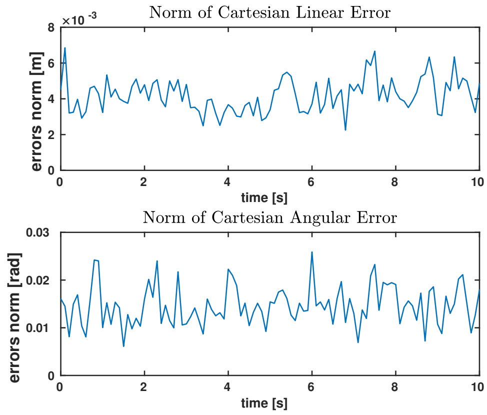
Mono (left Camera) Case



Mono (right Camera) Case



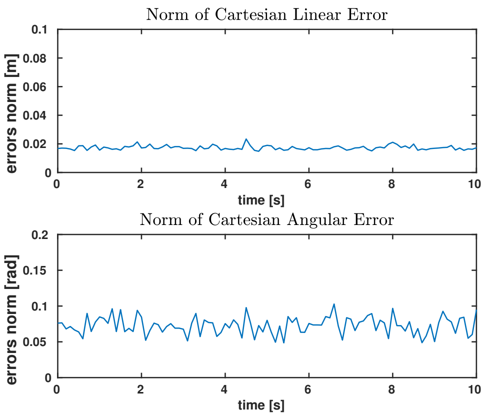
Stereo Camera Case



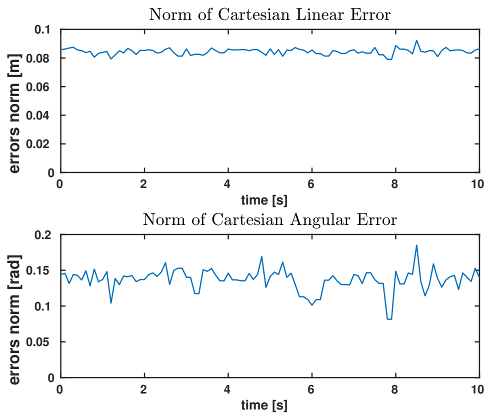
Stereo Depth Camera Case

Figure 4.5: Linear and Angular Error (in norm) between true pose and estimated pose. The detection step here used the find square method.

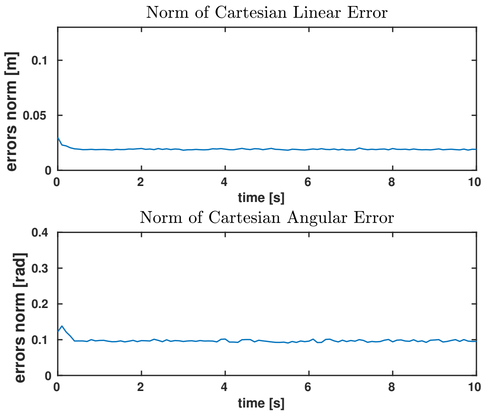
Template Matching Initialization



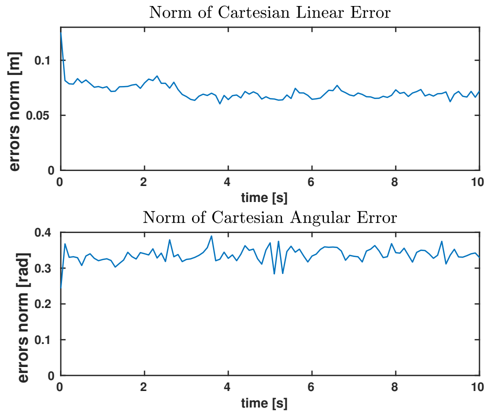
Mono (left Camera) Case



Mono (right Camera) Case



Stereo Camera Case



Stereo Depth Camera Case

Figure 4.6: Linear and Angular Error (in norm) between true pose and estimated pose. The detection step here used the template matching method.

Chapter 5

Conclusions

Write the conclusions here...

Appendix A

Other Algorithms for Object Detection

During the simulations, several trials have been done to find a suitable algorithm for the detection of the hole structure. In section (TODO) two methods have been discussed as the successful ones. In this appendix, others are briefly explained and discussed. Even if they are not used in the last versions of experiments, they can be useful for other purposes, such as detection of other kind of shapes. They can also be useful when the scene is different, for example while taking the hole structure from its side.
Each one is taken from OpenCV Detection tutorials, where also other algorithms can be found.

A.1 Corner Detection with the Shi-Harris method

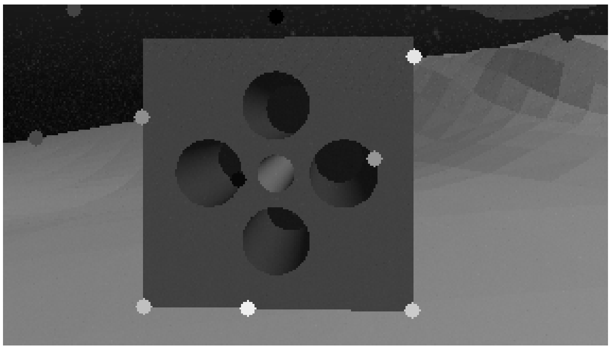


Figure A.1: goodFeaturesToTrack() result. Only two detected points are real corners, and the upper corners are not detected

Following the tutorial, I implemented a corner detector with the Shi-Harris method [Shi & Tomasi (2000)] using the OpenCV function goodFeaturesToTrack().
This function acts as corner detector. In our case, this can be useful to find corners of the square hole structure. These points can be used successively as starting point to higher level vision algorithms (e.g. draw the square shape to then understand its pose).
The original example lets change the number of maximum points to be found. This is useful to reduce the number of false positive corners. The main problem is that the real corners of the square are not the "best" ones. So we can't simply put this parameter equal to 4. On the other side, with bigger number of points, is then difficult to discriminate the right corners from the others.
Other interesting parameters are:

- **minDistance** The minimum distance between corners to be found.
- **qualityLevel** Parameter characterizing the minimal accepted quality of image corners.
- **blockSize** Size of an average block for computing a derivative covariation matrix over each pixel neighbourhood.
- **mask** To specify a certain region of interest in the image. In such a way, corners are found only in this region. The problem in our case is that without prior works we can't know where is the hole surface.

The points detected are effectively good feature points (as can be see in A.1). But the best ones, are not the ones that we want to detect (ie, the corner of the square).
This method should be used as a low level algorithm, to then help higher level ones. For example, to construct some polygons and to check if these polygons are square/rectangles. However, to follow this direction should be better to start from the edges. Another function can be to initialize a tracking by keypoints.

A.2 Canny Edge and Hough Transform

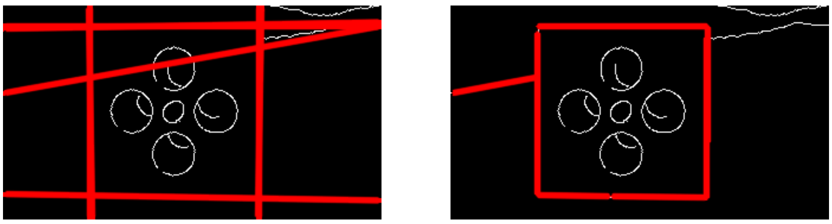


Figure A.2: Result of Hough Standard Transform (left) and Probabilistic (right). In white all the edges detected with Canny, in red the detected straight lines

The Hough Transform [Duda & Hart (1972)] is a method to detect straight lines in an image. Usually, a preprocessing of the image with an edge detector is used to improve the results, for example with a Canny Edge Detector [Canny (1986)]. The OpenCV tutorial makes use of two types of Hough Transform: the standard *HoughLines()* and the probabilistic *HoughLinesP()* [Matas *et al.* (2000)].
Results are visible in A.2. The probabilistic method shows that this function is good to detect the square structure of the hole. Thus, this method can be used as good starting point to further process the image, to then estimate the pose of the hole, or of other objects of interest.

A.3 Bounding Boxes Detection

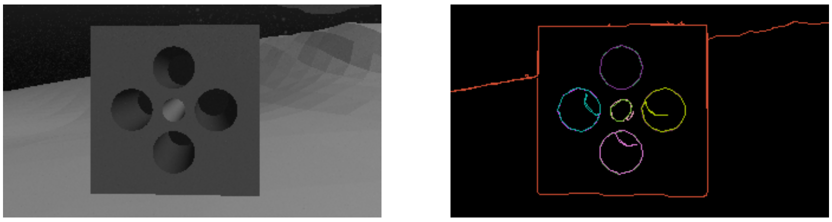


Figure A.3: Result of the algorithm: on the left the original image, on the right the contours detected

Figure A.4: On the left, the original image. On the right, the drawn bounding boxes around the holes.

In this method, shape contours are found and then bounding boxes drawn. As explained in the previous section, finding shape contours can be a starting point to initialize higher level image processing, such as the tracking.
The code derived from an OpenCV tutorial.
First, a Canny edge detector is used to preprocess the image. Then, the function *findContours()* is called to retrieves contours with the algorithm described in Suzuki & Be (1985). Finally, rectangles are draw around as a bounding boxes.
In case of the square hole structure (already almost a "bounding box" of itself), finding the boxes is useful to reduce the noise, and to have a square/rectangle with straight lines.
The result without the bounding boxes are shown in A.3. The square is noticeable, but also other not interesting lines are shown. For this specific purpose (detection of square face of hole), the detection is worse than the algorithm of section A.2 which used probabilistic hough transform.
The thing to notice is that the holes on the surface are well visible. This may be useful for other algorithms, or to detect other kind of shapes like a tube hole (a

pipe).
Results in A.4 show bounding boxes around the holes. However, it is difficult to have a precise pose estimation with bounding boxes.
In conclusion, this is a good method that can be explored. However, lot of non interesting edge are detected, so parameters have to be chosen wisely.

A.4 2D Feature Matching & Homography

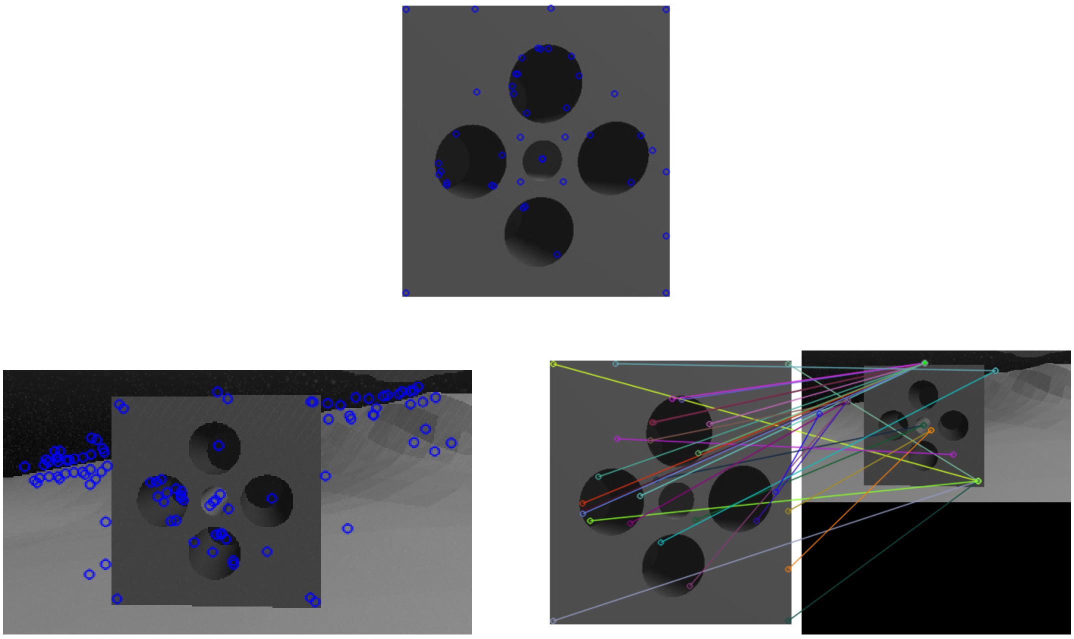


Figure A.5: Result of the algorithm. Above, the detected features (in blue) in the object image. Below, on the left, the detected features in the scene image; on the right the matched (erroneous) features.

Image features are small patches that are useful to compute similarities between images. Please note that these features are different from corner points. They indicates particular details that are different from others. Detecting these areas is important to recognize objects of interest in the image. The *descriptors* of these features contain the visual description of the patches, and they are used to recognize similarities.
This method uses a *object image* and a *scene image*. The first is a sort of template

(note that this method is not a template matching (TODO CITA MY SEZ)) which contains only the object to be found (in this case, the square face of the hole). The second is the image in which we want to detect this object (in this case, what the camera is seeing).
After good *features* are extracted from both images, the *descriptors* are used to match them, thus, detecting the object in the scene. Then, it is necessary to find the perspective transformation between object image and the scene (i.e. find homography). This is needed to take into account that usually pose and scaling of the object in the scene are not the same of the object image.
The OpenCV tutorial use different tools:

- **SURF** (Speeded Up Robust Features) Detector [Bay *et al.* (2006)] to extract features from *object image* and *scene image*, and to compute descriptors.
- **FLANN** (Fast Library for Approximate Nearest Neighbors) matcher [Muja & Lowe (2012)] to match the features.
- **Lowe's ratio test** [Lowe (2004)]to filter the best matches.
- **RANSAC** (RANdom SAmple Consensus) [Fischler & Bolles (1981)] method to find the homography with the function *findHomography()*.

In this case, results are unsatisfactory as can be seen in A.5. The main problem is that in this particular scene there are not nice distinct features. Also, the symmetry of the structure does not help, because there are a lot of particulars that are the same (like the square side and the holes). As can be seen in the tutorial, good results are obtained for food boxes. In fact, this methods is often associate to scenes where a lot of details are present (graffiti painting, supermarket shelf, ...). In our underwater case, realistic infrastructures don't have this details.
There are also lot of parameters to set for the three main tools (SURF, FLANN, RANSAC). Various trials have been tried but no-one was satisfactory. Also, different detectors (like SWIFT [Lowe (2004)]) and matchers (like Brute-force), have been tried.
The variety of tools and parameters make this method suitable for a lot of applications, and must be taken into consideration in other applications.

References

ABDULLAH, M., ROTH, H., WEYRICH, M. & WAHRBURG, J. (2015). An approach for peg-in-hole assembling using intuitive search algorithm based on human behavior and carried by sensors guided industrial robot. *IFAC-PapersOnLine*, **48**, 1476–1481. 7

ABREU, P., MORISHITA, H., PASCOAL, A., RIBEIRO, J. & SILVA, H. (2016). Marine Vehicles with Streamers for Geotechnical Surveys: Modeling, Positioning, and Control. *IFAC-PapersOnLine*. 25

ANTONELLI, G. (2009). Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems. *IEEE Transactions on Robotics*, **25**, 985–994. 5

ANTONELLI, G. & CHIAVERINI, S. (1998). Task-priority redundancy resolution for underwater vehicle-manipulator systems. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 1, 768–773 vol.1. 5

ANTONELLI, G. & CHIAVERINI, S. (2003). Fuzzy redundancy resolution and motion coordination for underwater vehicle-manipulator systems. *IEEE Transactions on Fuzzy Systems*, **11**, 109–120. 5

ANTONELLI, G., ARRICHIELLO, F. & CHIAVERINI, S. (2008). The null-space-based behavioral control for autonomous robotic systems. *Intelligent Service Robotics*, **1**, 27–39. 5

ANTONELLI, G., INDIVERI, G. & CHIAVERINI, S. (2009). Prioritized closed-loop inverse kinematic algorithms for redundant robotic systems with velocity saturations. 5892–5897. 15

BAERLOCHER, P. & BOULIC, R. (2004). An inverse kinematics architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer*, **20**, 402–417. 7

BAY, H., TUYTELAARS, T. & VAN GOOL, L. (2006). Surf: Speeded up robust features. In A. Leonardis, H. Bischof & A. Pinz, eds., *Computer Vision – ECCV 2006*, 404–417, Springer Berlin Heidelberg, Berlin, Heidelberg. 47

BINGHAM, B., FOLEY, B., SINGH, H., CAMILLI, R., DELAPORTA, K., EUSTICE, R., MALLIOS, A., MINDELL, D., ROMAN, C. & SAKELLARIOU, D. (2010). Robotic tools for deep water archaeology: Surveying an ancient shipwreck with an autonomous underwater vehicle. *Journal of Field Robotics*, **27**, 702–717. 2

BRADSKI, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*. 29

CANNY, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-8**, 679–698. 44

CAPOCCI, R., DOOLY, G., OMERDIC, E., COLEMAN, J., NEWE, T. & TOAL, D. (2017). Inspection-class remotely operated vehiclesâĂŤa review. *Journal of Marine Science and Engineering*, **5**, 13. 2

CARRERA, A., PALOMERAS, N., HURTOS, N., KORMUSHEV, P. & CARRERAS, M. (2014). Learning by demonstration applied to underwater intervention. vol. 269. 4

CASALINO, G., ANGELETTI, D., BOZZO, T. & MARANI, G. (2001). Dexterous underwater object manipulation via multi-robot cooperating systems. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, vol. 4, 3220–3225 vol.4. 3

CASALINO, G., ANGELETTI, D., CANNATA, G. & MARANI, G. (2002). The functional and algorithmic design of amadeus multirobot workcell. In S.K. Choi & J. Yuh, eds., *Underwater Vehicle Technology*, vol. 12. 3

CASALINO, G., TURETTA, A., SORBARA, A. & SIMETTI, E. (2009). Self-organizing control of reconfigurable manipulators: A distributed dynamic programming based approach. In *2009 ASME/IFToMM International Conference on Reconfigurable Mechanisms and Robots*, 632–640. 5

CASALINO, G., ZEREIK, E., SIMETTI, E., TORELLI, S., SPERINDÉ, A. & TURETTA, A. (2012). Agility for underwater floating manipulation task and subsystem priority based control strategy. In *International Conference on Intelligent Robots and Systems (IROS 2012)*, 1772–1779. 4

CASALINO, G., CACCIA, M., CAITI, A., ANTONELLI, G., INDIVERI, G., MELCHIORRI, C. & CASELLI, S. (2014). Maris: A national project on marine robotics for interventions. In *22nd Mediterranean Conference on Control and Automation*, 864–869. 4, 7

CHANG, R.J., Y. LIN, C. & S. LIN, P. (2011). Visual-based automation of peg-in-hole microassembly process. *Journal of Manufacturing Science and Engineering*, **133**, 041015. 7

CHENG CHANG, C., YUAN CHANG, C. & TING CHENG, Y. (2004). Distance measurement technology development at remotely teleoperated robotic manipulator system for underwater constructions. 333 – 338. 2

CHHATPAR, S.R. & BRANICKY, M.S. (2001). Search strategies for peg-in-hole assemblies with position uncertainty. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, vol. 3, 1465–1470 vol.3. 8

CHIAVERINI, S. (1997). Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Transactions on Robotics and Automation*, **13**, 398–410. 5

CHRIST, R. & WERNLI, R. (2013). *The ROV Manual: A User Guide for Remotely Operated Vehicles*. Elsevier, 2nd edn. 2

CIESLAK, P., RIDAO, P. & GIERGIEL, M. (2015). Autonomous underwater panel operation by girona500 uvms: A practical approach to autonomous underwater manipulation. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 529–536. 4

COMPORT, A., MARCHAND, E., PRESSIGOUT, M. & CHAUMETTE, F. (2006). Real-time markerless tracking for augmented reality: the virtual visual servoing framework. *IEEE Transactions on Visualization and Computer Graphics*, **12**, 615–628. 34

COOK, D., VARDY, A. & LEWIS, R. (2014). A survey of auv and robot simulators for multi-vehicle operations. In *2014 IEEE/OES Autonomous Underwater Vehicles (AUV)*, 1–8. 23

DI LILLO, P.A., SIMETTI, E., DE PALMA, D., CATALDI, E., INDIVERI, G., ANTONELLI, G. & CASALINO, G. (2016). Advanced rov autonomy for efficient remote control in the dexrov project. *Marine Technology Society Journal*, **50**. 4

DIAZ LEDEZMA, F., AMER, A., ABDELLATIF, F., OUTA, A., TRIGUI, H., PATEL, S. & BINYAHIB, R. (2015). A market survey of offshore underwater robotic inspection technologies for the oil and gas industry. 2

DIETRICH, F., BUCHHOLZ, D., WOBBE, F., SOWINSKI, F., RAATZ, A., SCHUMACHER, W. & WAHL, F.M. (2010). On contact models for assembly tasks: Experimental investigation beyond the peg-in-hole problem on the example of force-torque maps. In

2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2313–2318. 7

DJAPIC, V., NAÄŚ, Ä., FERRI, G., OMERDIC, E., DOOLY, G., TOAL, D. & VUKIÄĞ, Z. (2013). Novel method for underwater navigation aiding using a companion underwater robot as a guiding platforms. In *2013 MTS/IEEE OCEANS - Bergen*, 1–10. 2

DRAP, P. (2012). Underwater photogrammetry for archaeology. *Special Applications of Photogrammetry*. 2

DUDA, R.O. & HART, P.E. (1972). Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, **15**, 11–15. 44

E. ROHMER, M.F., S. P. N. SINGH (2013). V-rep: a versatile and scalable robot simulation framework. In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*. 23

ESCANDE, A., MANSARD, N. & WIEBER, P.B. (2014). Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *I. J. Robotics Res.*, **33**, 1006–1028. 7

EVANS, J., REDMOND, P., PLAKAS, C., HAMILTON, K. & LANE, D. (2003). Autonomous docking for intervention-auvs using sonar and video-based real-time 3d pose estimation. In *Oceans 2003. Celebrating the Past ... Teaming Toward the Future (IEEE Cat. No.03CH37492)*, vol. 4, 2201–2210 Vol.4. 3

EVANS, J.C., KELLER, K.M., SMITH, J.S., MARTY, P. & RIGAUD, O.V. (2001). Docking techniques and evaluation trials of the swimmer auv: an autonomous deployment auv for work-class rovs. In *MTS/IEEE Oceans 2001. An Ocean Odyssey. Conference Proceedings (IEEE Cat. No.01CH37295)*, vol. 1, 520–528 vol.1. 3

FAVERJON, B. & TOURNASSOUD, P. (1987). A local based approach for path planning of manipulators with a high number of degrees of freedom. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, vol. 4, 1152–1159. 6

FISCHLER, M.A. & BOLLES, R.C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, **24**, 381–395. 47

FLACCO, F., DE LUCA, A. & KHATIB, O. (2012). Prioritized multi-task motion control of redundant robots under hard joint constraints. 3970–3977. 5

FLETCHER, B. (2000). Worldwide undersea mcm vehicle technologies. 10. 2

GILMOUR, B., NICCUM, G. & O'DONNELL, T. (2012). Field resident auv systems âĂŤ chevron's long-term goal for auv development. In *2012 IEEE/OES Autonomous Underwater Vehicles (AUV)*, 1–5. 2

GUENNEBAUD, G., JACOB, B. *et al.* (2010). Eigen v3 [online; accessed 29-june-2019]. 26

KANOUN, O., LAMIRAUX, F. & WIEBER, P. (2011). Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task. *IEEE Transactions on Robotics*, **27**, 785–792. 6

KERMORGANT, O. (2014). A dynamic simulator for underwater vehicle-manipulators. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots Simpar*, Springer, Bergamo, Italy. 23

KHATIB, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, **5**, 90–98. 5

KHATIB, O. (1987). A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, **3**, 43–53. 5

LANE, D.M., DAVIES, J.B.C., CASALINO, G., BARTOLINI, G., CANNATA, G., VERUGGIO, G., CANALS, M., SMITH, C., O'BRIEN, D.J., PICKETT, M., ROBINSON, G., JONES, D., SCOTT, E., FERRARA, A., ANGELLETI, D., COCCOLI, M., BONO, R., VIRGILI, P., PALLAS, R. & GRACIA, E. (1997). Amadeus: advanced manipulation for deep underwater sampling. *IEEE Robotics Automation Magazine*, **4**, 34–45. 3

LANE, D.M., MAURELLI, F., KORMUSHEV, P., CARRERAS, M., FOX, M. & KYRIAKOPOULOS, K. (2012). Persistent autonomy: the challenges of the pandora project. *IFAC Proceedings Volumes*, **45**, 268 – 273, 9th IFAC Conference on Manoeuvring and Control of Marine Craft. 4

LEE, H. & PARK, J. (2014). An active sensing strategy for contact location without tactile sensors using robot geometry and kinematics. *Autonomous Robots*, **36**, 109–121. 8

LEE, J., MANSARD, N. & PARK, J. (2012). Intermediate desired value approach for task transition of robots in kinematic control. *IEEE Transactions on Robotics*, **28**, 1260–1277. 6

LOWE, D.G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, **60**, 91–110. 47

LOZANO-PÃĽREZ, T., MASON, M.T. & TAYLOR, R.H. (1984). Automatic synthesis of fine-motion strategies for robots. *The International Journal of Robotics Research*, **3**, 3–24. 8

MACIEJEWSKI, A.A. & KLEIN, C.A. (1985). Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *The International Journal of Robotics Research*, **4**, 109–117. 5

MANSARD, N., KHATIB, O. & KHEDDAR, O. (2009a). A unified approach to integrate unilateral constraints in the stack of tasks. *IEEE Transactions on Robotics*, **25**, 670–685. 6

MANSARD, N., REMAZEILLES, A. & CHAUMETTE, F. (2009b). Continuity of varying-feature-set control laws. *IEEE Transactions on Automatic Control*, **54**, 2493–2505. 6

MARANI, G., KIM, J., YUH, J. & CHUNG, W. (2003). Algorithmic singularities avoidance in task-priority based controller for redundant manipulators. vol. 4, 3570 – 3574 vol.3. 5

MARANI, G., CHOI, S.K. & YUH, J. (2009). Underwater autonomous manipulation for intervention missions auvs. *Ocean Engineering*, **36**, 15 – 23, autonomous Underwater Vehicles. 3

MARCHAND, E., SPINDLER, F. & CHAUMETTE, F. (2005). Visp for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics and Automation Magazine*, **12**, 40–52. 29

MARTY, P. (2004). Alive: An autonomous light intervention vehicle. *Scandinavian Oil-Gas Magazine*, **32**. 3

MATAS, J., GALAMBOS, C. & KITTLER, J. (2000). Robust detection of lines using the progressive probabilistic hough transform. *Computer Vision and Image Understanding*, **78**, 119–137. 44

MICHEL, O. (2004). Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems*, **1**, 39–42. 23

MIURA, J. & IKEUCHI, K. (1998). Task-oriented generation of visual sensing strategies in assembly tasks. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **20**, 126 – 138. 7

MUJA, M. & LOWE, D.G. (2012). Fast matching of binary features. In *Computer and Robot Vision (CRV)*, 404–410. 47

NAKAMURA, Y. (1990). *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edn. 5

NAKAMURA, Y. & HANAFUSA, H. (1986). Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement, and Control*. 5

NENCHEV, D.N. & SOTIROV, Z.M. (1994). Dynamic task-priority allocation for kinematically redundant robotic mechanisms. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*, vol. 1, 518–524 vol.1. 6

NEWMAN, W., ZHAO, Y. & PAO, Y.H. (2001). Interpretation of force and moment signals for compliant peg-in-hole assembly. vol. 1, 571 – 576 vol.1. 7

OH, J. & OH, J.H. (2015). A modified perturbation/correlation method for force-guided assembly. *Journal of Mechanical Science and Technology*, **29**, 5437–5446. 8

PADIR, T. (2005). Kinematic redundancy resolution for two cooperating underwater vehicles with on-board manipulators. vol. 4, 3137 – 3142 Vol. 4. 5

PARAVISI, M., H. SANTOS, D., JORGE, V., HECK, G., GONÃĞALVES, L.M. & AMORY, A. (2019). Unmanned surface vehicle simulator with realistic environmental disturbances. *Sensors*, **19**. 23, 24

PARK, H., BAE, J.H., PARK, J.H., BAEG, M.H. & PARK, J. (2013). Intuitive peg-in-hole assembly strategy with a compliant manipulator. In *IEEE ISR 2013*, 1–5. 8

PARK, H., PARK, J., LEE, D.H., PARK, J.H., BAEG, M.H. & BAE, J.H. (2017). Compliance-based robotic peg-in-hole assembly strategy without force feedback. *IEEE Transactions on Industrial Electronics*, **PP**, 1–1. 8

PAULI, J., SCHMIDT, A. & SOMMER, G. (2001). Vision-based integrated system for object inspection and handling. *Robotics and Autonomous Systems*, **37**, 297 – 309. 7

PEREZ, T. & FOSSEN, T.I. (2007). Kinematic models for manoeuvring and seakeeping of marine vessels. *Modeling, Identification and Control*, **28**, 19–30. 11

PRATS, M., PÃĽREZ, J., FERNÃĄNDEZ, J.J. & SANZ, P.J. (2012). An open source tool for simulation and supervision of underwater intervention missions. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2577–2582. 22

PRATS, M., ROMAGÃŞS, D., PALOMERAS, N., GARCÃ A SÃĄNCHEZ, J.C., NANNEN, V., WIRTH, S., FERNANDEZ, J., P. BELTRÃĄN, J., CAMPOS, R., RIDAO, P., SANZ, P., OLIVER, G., CARRERAS, M., GRACIAS, N., MARÃ N PRADES, R. & ORTIZ, A. (2012). Reconfigurable auv for intervention missions: A case study on underwater object recovery. *Intelligent Service Robotics*, **5**, 19–31. 3

PRESSIGOUT, M. & MARCHAND, E. (2007). Real-time hybrid tracking using edge and texture information. *The International Journal of Robotics Research*, **26**, 689–713. 34

PROMETEO (2016). http://www.irs.uji.es/prometeo/, [online; accessed 25-october-2018]. 4

RIGAUD, V., COSTE-MANIERE, E., ALDON, M.J., PROBERT, P., PERRIER, M., RIVES, P., SIMON, D., LANG, D., KIENER, J., CASAL, A., AMAR, J., DAUCHEZ, P. & CHANTLER, M. (1998). Union: underwater intelligent operation and navigation. *IEEE Robotics Automation Magazine*, **5**, 25–35. 3

ROBUST (2016). http://eu-robust.eu, [online; accessed 25-october-2018]. 4

RUSU, R.B. & COUSINS, S. (2011). 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China. 29, 35

SANZ, P., RIDAO, P., OLIVER, G., CASALINO, G., INSAURRALDE, C., SILVESTRE, C., MELCHIORRI, C. & TURETTA, A. (2012). Trident: Recent improvements about autonomous underwater intervention missions. vol. 3, 1–10. 3, 7

SCHEMPF, H. & YOERGER, D.R. (1992). Coordinated vehicle/manipulation design and control issues for underwater telemanipulation. *IFAC Proceedings Volumes*, **25**, 259 – 267, iFAC Workshop on Artificial Intelligence Control and Advanced Technology in Marine Automation (CAMS '92), Genova, Italy, April 8-10. 3

SENTIS, L. & KHATIB, O. (2005). Control of free-floating humanoid robots through task prioritization. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 1718–1723. 5

SHI, J. & TOMASI, C. (2000). Good features to track. *Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, **600**, 593âĂŞ600. 43

SHIRINZADEH, B., ZHONG, Y., TILAKARATNA, P.D.W., TIAN, Y. & DALVAND, M.M. (2011). A hybrid contact state analysis methodology for robotic-based adjustment of cylindrical pair. *The International Journal of Advanced Manufacturing Technology*, **52**, 329–342. 7

SICILIANO, B. & SLOTINE, J..E. (1991). A general framework for managing multiple tasks in highly redundant robotic systems. In *Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments*, 1211–1216 vol.2. 5

SICILIANO, B., SCIAVICCO, L., VILLANI, L. & ORIOLO, G. (2009). *Robotics: Modelling, Planning and Control*, 147–151. Springer-Verlag London. 20

SIMETTI, E. & CASALINO, G. (2016). A novel practical technique to integrate inequality control objectives and task transitions in priority based control. *Journal of Intelligent & Robotic Systems*, **84**. 4, 7, 14, 15, 26

SIMETTI, E. & CASALINO, G. (2017). Manipulation and transportation with cooperative underwater vehicle manipulator systems. *IEEE Journal of Oceanic Engineering*, **42**, 782–799. 4, 10, 16, 24, 25

SIMETTI, E., TURETTA, A. & CASALINO, G. (2009). *Distributed Control and Coordination of Cooperative Mobile Manipulator Systems*, 315–324. Springer Berlin Heidelberg, Berlin, Heidelberg. 5

SIMETTI, E., CASALINO, G., TORELLI, S., SPERINDÉ, A. & TURETTA, A. (2014a). Floating underwater manipulation: Developed control methodology and experimental validation within the trident project. *Journal of Field Robotics*, **31(3)**, 364–385. 4, 7

SIMETTI, E., CASALINO, G., TORELLI, S., SPERINDÉ, A. & TURETTA, A. (2014b). Underwater floating manipulation for robotic interventions. *IFAC Proceedings Volumes*, **47**, 3358 – 3363, 19th IFAC World Congress. 7

SIMETTI, E., CASALINO, G., WANDERLINGH, F. & AICARDI, M. (2018). Task priority control of underwater intervention systems: Theory and applications. *Ocean Engineering*, **164**, 40 – 54. 4, 10

SONG, H., KIM, Y. & SONG, J.B. (2016). Guidance algorithm for complex-shape peg-in-hole strategy based on geometrical information and force control. *Advanced Robotics*, 1–12. 8

SUGIURA, H., GIENGER, M., JANSSEN, H. & GOERICK, C. (2007). Real-time collision avoidance with whole body motion control for humanoid robots. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2053–2058. 5

SUZUKI, S. & BE, K.A. (1985). Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, **30**, 32 – 46. 31, 45

TRINH, S., SPINDLER, F., MARCHAND, E. & CHAUMETTE, F. (2018). A modular framework for model-based visual tracking using edge, texture and depth features. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'18*, Madrid, Spain. 34

TWINBOT (2019). http://www.irs.uji.es/twinbot/twinbot.html, [online; accessed 29-june-2019]. 1, 8, 28

URABE, T., URA, T., TSUJIMOTO, T. & HOTTA, H. (2015). Next-generation technology for ocean resources exploration (zipangu-in-the-ocean) project in japan. 1–5. 2

WANDERLINGH, F. (2018). *Cooperative Robotic Manipulation for the Smart Factory*. Ph.D. thesis, Università degli Studi di Genova. 10, 19

WYNN, R., HUVENNE, V., LE BAS, T., MURTON, B., CONNELLY, D., BETT, B., RUHL, H., MORRIS, K., PEAKALL, J., PARSONS, D., J. SUMNER, E., E. DARBY, S., DORRELL, R. & HUNT, J. (2014). Autonomous underwater vehicles (auvs): Their past, present and future contributions to the advancement of marine geoscience. *Marine Geology*, **352**. 2

XU, Q. (2015). Robust impedance control of a compliant microgripper for high-speed position/force regulation. *IEEE Transactions on Industrial Electronics*, **62**, 1201–1209. 8

YOSHIKAWA, T. (1984). Analysis and Control of Robot Manipulators with Redundancy. In M. Brady & R. Paul, eds., *Robotics Research The First International Symposium*, 735–747, MIT Press. 5

YUH, J., CHOI, S.K., IKEHARA, C., KIM, G.H., MCMURTY, G., GHASEMI-NEJHAD, M., SARKAR, N. & SUGIHARA, K. (1998). Design of a semi-autonomous underwater vehicle for intervention missions (sauvim). In *Proceedings of 1998 International Symposium on Underwater Technology*, 63–68. 3

ZEREIK, E., SORBARA, A., MERLO, A., SIMETTI, E., CASALINO, G. & DIDOT, F. (2011). Space robotics supporting exploration missions: vision, force control and coordination strategy for crew assistants. *Intelligent Service Robotics*, **4**, 39–60. 5