Contributions to Motion Parameterization, Planning and Control for Robotic Systems

Adrian Burlacu

Department of Automatic Control and Applied Informatics
Faculty of Automatic Control and Computer Engineering
Gheorghe Asachi Technical University of Iasi

This dissertation is submitted for the degree of
Habilitation Thesis

March 2020
Abstract

The activities and results presented in this Habilitation Thesis cover some of the most significant scientific and professional achievements subsequent to February 2009, when my PhD thesis was defended. These achievements have been obtained during my academic and research career at the Faculty of Automation Control and Computer Engineering, Gheorghe Asachi Technical University of Iasi. Currently, I am Associate Professor in the Department of Automatic Control and Applied Informatics of this faculty.

In the last 10 years, since the PhD thesis was defended, I was the director of 2 national research projects, coordinator of a research project with industrial beneficiary, member in 2 international research projects (H2020 and Brancusi Grant), member of the research teams in 4 national. I also made contributions to the development of the faculty research infrastructure (equipment and research laboratory platforms) and I am currently involved as a main researcher in the Research Laboratory of Robotics and Intelligent Systems (LoRIS).

My research expertise is related to advanced robot motion control and its applications. The publication activity since 2009 can be summarized as: 6 monographs, manuals and chapters; over 20 articles in indexed journals (14 indexed in Clarivate Analytics Web of Science with an Impact factor of over 27); over 50 papers in IEEE, IFAC and other relevant conference proceedings (27 indexed in Clarivate Analytics Web of Science). All the papers were citated in more than 300 publications (WoS citations 167 h-index 7; Scopus citations 325 h-index 9; GScholar citations 463 h-index 12). An overview on my professional evolution and scientific results and future career developments are presented in the last two chapters of the thesis.

Chapter 2 describes the current status of the fields of interest along with the evolution until now and future trends emphasizing the importance of motion parameterization, planning and control in robotic systems, together with the difficulties and limitations of current approaches.

Chapter 3 presents the main achievements in the area of motion parameterization using dual Lie algebras. Rigid-body displacement and motion parameterization can be done using multiple algebraic entities. When choosing a parameterization method, a very important criterion is the number of algebraic equations and variables. Recently, orthogonal dual tensors proved to be a complete free of coordinates tool for computing rigid body displacement and
motion parameters. The chapter gives details on the properties of the isomorphism between the special Euclidean group and the orthogonal dual tensors group. This isomorphism allows the development of new techniques for solving different kinematic problems. After a sub-chapter which provides the necessary mathematical preliminaries, the techniques based on orthogonal dual tensors for rigid body motion parameterization are discussed. In the end are presented solutions for two specific kinematic problems: motion parameterization from feature-based description of a rigid body and $AX = XB$ sensor calibration problem.

Chapter 4 presents the research on path planning for multi agent systems to solve different capacity and energy constrained vehicle routing problems. For such problems, various mixed-integer linear programming formulations are given. The main goal is to plan a team of mobile agents such that they gather samples scattered throughout an environment into single or multiple storage facilities. The problem’s hypothesis consists in a team of mobile agents which must bring to a deposit region a set of samples that exist in an environment at known locations. Throughout the chapter, new mathematical models are presented for the considered problems and both numerical evaluation by simulation and real-time experiments are discussed. The chapter is structured in three parts: research area context, solutions for sample gathering problems and applications. The applications include numerical evaluation of different solutions for path planning for a team of mobile with wheels and a real-time experiment for a team of autonomous drones.

Chapter 5 focuses on the main contributions in visual based perception and control architectures for robotic systems. The chapter is structured in two parts. First, techniques and results for disparity based perception in 3D environments are presented. Free space and obstacle detection represent two of the most active research areas that involve active or passive sensors. Both are related with an accurate ground area identification. Stereo vision systems are passive sensors that can be used to evaluate data and measure distances in front of the camera. Multiple areas, such as robotics, automotive and assistive systems, benefit from research results involving stereo vision cameras. The second part of the chapter is dedicated to advanced visual based control techniques. The use of visual sensors for closing the feedback control loop is an important approach in multiple applications. Visual servoing architectures can be designed using different control techniques: image based (IBVS), 3D data based (PBVS), 2 1/2D, hybrid, partitioned or switched. The IBVS and PBVS schemes are the most known techniques for the use of computer vision data in robot motion control. In this part original research on visual predictive control (VPC) are discussed. The applications include the control of an industrial manipulator using a VPC architecture.
Activitățile și rezultatele din această teză teză de abilitare prezintă cele mai importante realizări științifice și profesionale ulterioare lunii februarie 2009, data prezentării tezei de doctorat. Aceste realizări au fost obținute în perioada evoluției mele în cadrul Facultății de Automatică și Calculatoare, Universitatea Tehnică Gheorghe Asachi din Iași. În prezent, sunt conferențiar în cadrul departamentului de Automatică și Informatică Aplicată.

În ultimii 10 ani am fost coordonator a două proiecte de cercetare naționale, coordonator a unui proiect de cercetare cu beneficiar industrial, membru în două proiecte de cercetare internaționale (H2020 și un program bilateral Brâncuși), membru în 5 proiecte naționale de cercetare. În același timp am contribuit la modernizarea infrastructurii de cercetare din cadrul facultății, momentan fiind membru al laboratorului de cercetare de robotică și sisteme inteligente.

Experiența acumulată include competențe în domenii strâns legate de controlul mișcării sistemelor robotizate și aplicațiilor interdisciplinare. Activitatea de diseminare începând cu 2009 poate fi descrisă prin: 5 cărți, manuale didactice sau capitole de carte; peste 20 de articole în reviste indexate în baze de date (14 indexate în Clarivate Analytics Web of Science având un factor de impact cumulat mai mare de 27); peste 50 de lucrări în IEEE, IFAC sau în alte volume indexate ale conferințelor internaționale (27 indexate în Clarivate Analytics Web of Science). Publicațiile au fost citate în peste 300 de articole (WoS 167 h-index 7; Scopus 325 h-index 9; GScholar 463 h-index 12). Detalii suplimentare despre evoluția profesională, rezultatele cercetărilor și direcții de dezvoltare a carierei sunt prezentate în ultimele două capitole ale tezei.

Capitolul 2 prezintă stadiul curent al domeniului de interes, fiind scos în evidență evoluția și tendințele viitoare privind parametrizarea, planificarea și controlul mișcării sistemelor robotizate precum și limitările abordărilor existente.

Capitolul 3 prezintă principalele realizări în domeniul descrierii mișcărilor rigide folosind algebre Lie duale. Postura și mișcarea corpurilor rigide pot fi descrise folosind diferite tipuri de parametrizări algebrice. Alegerea unei metode de parametrizare depinde de mai multe criterii, unul dintre acestea fiind numărul de variabile și ecuații utilizat. Recent a fost demonstrat faptul că grupul tensorilor ortogonali duali reprezintă o alternativă invariantă la sisteme de coordonate pentru descrierea parametrilor posturii și mișcării corpurilor rigide. În acest capitol sunt analizate principalele proprietăți ale izomorfismului dintre grupul special Euclidian și grupul tensorilor ortogonali duali. Acest izomorfism permite dezvoltare unor soluții noi pentru diferite probleme de cinematică. Capitolul conține preliminariile matematice necesare dezvoltării soluțiilor de parametrizare a mișcării rigide folosind tensori ortogonali duali. În finalul capitolului sunt prezentate soluții noi pentru două probleme
specifice de cinematică: parametrizarea miscării rigide folosind descrierea prin trăsături a corpurilor rigide și problema \( AX = XB \) de calibrare a senzorilor.

Capitolul 4 prezintă cercetările privind planificarea traiectoriei pentru sistemele cu mai mulți agenți pentru a rezolva diferite probleme de rutare a vehiculelor cu capacitate și energie constrânsă. Pentru astfel de probleme, sunt prezentate diverse formulări de programare lineare mixte întregi. Scopul principal este de a planifica o echipă de agenți mobili, astfel încât să adune mostre împrăștiate într-un mediu în spații de stocare simple sau multiple. Ipoteza problemei constă într-o echipă de agenți mobili care trebuie să aducă într-o regiune de depozit un set de probe care există într-un mediu în locații cunoscute. Pe parcursul capitolului sunt prezentate modele matematice noi, se analizează atât performantele numerice prin simulare și folosind experimente în timp real. Capitolul este structurat în trei părți: contextul domeniului de cercetare, soluții pentru probleme de colectare a resurselor și aplicații. Aplicațiile includ evaluarea numerică a diferitelor soluții pentru planificarea traiectoriilor pentru o echipă de roboți mobili cu roti și un experiment în timp real pentru o echipă de drone autonome.

Capitolul 5 se concentrează pe contribuțiile principale în dezvoltarea arhitecturilor de percepție și control bazate pe senzori vizuali pentru sisteme robotizate. Capitolul este structurat în două părți. În prima parte, sunt prezentate tehnici și rezultate pentru percepția bazată pe disparitate în mediile 3D. Detectarea spațiului liber și a obstacolelor reprezintă două dintre cele mai importante teme de cercetare care implică senzori activi sau pasivi. Ambele sunt indiscutabil legate de o identificare precisă a căii de rulare. Sistemele de stereo vizuale sunt senzori pasivi care pot fi folosiți pentru evaluarea datelor și măsurarea distanțelor. Mai multe domenii, cum ar fi robotica sau sistemele auto de asistentă, beneficiază de rezultatele cercetării care implică camere stereo vizionate. A doua parte a capitolului este dedicată tehnicii avansate de control bazat pe vizual. Utilizarea senzorilor vizuali pentru închiderea buclei de control este o abordare importantă în multe aplicații ale sistemelor robotizate. Arhitecturile de feedback vizual pot fi proiectate folosind diferite tehnici de control: bazate pe imagini (IBVS), date 3D (PBVS), 2 1 / 2D, hibride, partiționate sau comutate. Schemele IBVS și PBVS sunt cele mai cunoscute tehnici pentru utilizarea datelor privind viziunea computerului în controlul miscării robotului. În această parte sunt discutate cercetările originale despre controlul predictiv vizual (VPC). Aplicațiile includ controlul unui manipulator industrial folosind o arhitectură VPC.
Table of contents

List of figures ix
List of tables xiii

1 State of the art and main contributions 1

2 Motion parameterization using dual Lie algebras 7
   2.1 Mathematical Preliminaries . . . . . . . . . . . . . . . . . . . . . . . . . 7
   2.2 Rigid body motion parameterization using orthogonal dual tensors . . . . 13
      2.2.1 Dual Tensor Construction using Dual Vectors Bases . . . . . . . . 13
      2.2.2 Isomorphism between $SE_3$ and $SO_3$ . . . . . . . . . . . . . . . . . . 22
   2.3 Applications . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28
      2.3.1 Recovering motion parameters from feature-based description of rigid body . . . . . . . . . . . . . . . . 28
      2.3.2 A different approach to solving the AX=XB sensor calibration problem 37

3 Path Planning for multi-robot task allocation problems 47
   3.1 Research context . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47
   3.2 Sample gathering problems and solutions . . . . . . . . . . . . . . . . . . 49
      3.2.1 Sample gathering using a team of mobile robots with wheels . . . . 49
      3.2.2 Sample gathering using a team of quadcopters . . . . . . . . . . . . 56
   3.3 Applications . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61
      3.3.1 Path planning for a team of mobile robots with wheels . . . . . . . 61
      3.3.2 Path Planning for a team of Autonomous Drones . . . . . . . . . . . 71

4 Perception and control using vision based techniques 77
   4.1 Disparity based perception of 3D environments . . . . . . . . . . . . . . . 77
      4.1.1 Obstacle detection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 80
      4.1.2 Ground geometry assessment . . . . . . . . . . . . . . . . . . . . . . . 86
4.2 Advanced visual based control techniques ........................................ 93
  4.2.1 Visual predictive control ....................................................... 93
  4.2.2 Nonlinear visual servoing systems using image moments ............ 102
  4.2.3 VPC architecture for industrial robots ................................... 108

5 Overview of professional and scientific results .................................. 115
  5.1 Education and training .............................................................. 115
  5.2 Professional .............................................................................. 116
  5.3 Teaching ................................................................................... 116
  5.4 Domains of competence and research activity ............................... 116
  5.5 Publications .............................................................................. 117
  5.6 Additional information ............................................................... 119

6 Future career development ................................................................. 121

References ......................................................................................... 125
List of figures

2.1 Geometrical interpretation of Theorem 1 ........................................... 14
2.2 Geometric representation of the dual rigid bases $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$ and $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ .... 25
2.3 Geometric representation of two sets of directed lines parameterized by dual vectors $\{ \mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n \}, \{ \mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_n \}$ which validate the rigidity constraints $\mathbf{a}_i \cdot \mathbf{a}_j = \mathbf{b}_i \cdot \mathbf{b}_j, \ i, j = 1, n$ ......................................................... 28
2.4 Rigid body features: points and lines. Combinations of these measurements generate dual rigid bases ................................................................. 29

3.1 Example: environment graph with 10 nodes, 3 robots and 14 samples. The robot moving costs are marked on graph edges, and the samples are represented by the blue discs. .................................................. 52
3.2 Success rates of MILP (a) and QP (b) vs. number of robots $N_R$ and number of samples $N_S$ ................................................................. 67
3.3 MILP (3.4): 2D projection for failures, defined as success rate of less than 50%. ................................................................. 69
3.4 Average optimization times, for $(N_R, N_S)$ pairs for which at least 5 feasible solutions from the 50 tests were obtained: (a) MILP (3.4), (b) QP (3.12), (c) IH from Alg. 6 ................................................................. 70
3.5 Differences between costs induced by the three solutions: (a) difference between QP and MILP costs, related to the optimal MILP cost; (b) difference between IH and QP suboptimal costs, related to QP cost ................................................................. 70
3.6 Decision diagram for choosing an appropriate solution for Problem 3.2.1 ................................................................. 70
3.7 Projection of a 3D environment, with two storage areas (green and red circles) and four delivery zones (cyan squares). The number of goods to be delivered in written on each storage area ................................................................. 71
3.8 Sketch of movement and delivery sequences for each drone. The green drone has 2 tours (the first represented with continuous lines, and the second with dashed ones), the red drone suffices one tour ................................................................. 72
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.9</td>
<td>Experimental platform</td>
</tr>
<tr>
<td>3.10</td>
<td>Parrot Mambo Drone and green goods</td>
</tr>
<tr>
<td>3.11</td>
<td>Real-time setup. Frame from a movie of the real-time experiments. The Kinect picture in on top, the side-view on bottom, the internal drone information on left. The delivery areas are on top of the three tall shapes, and the storage areas are the two colored “X” (green and red) on the ground.</td>
</tr>
<tr>
<td>4.1</td>
<td>The steps performed in the obstacles detection framework</td>
</tr>
<tr>
<td>4.2</td>
<td>Experiments with the KITTI dataset: Left image; Disparity map; Ground plane mask</td>
</tr>
<tr>
<td>4.3</td>
<td>U-disparity and U-disparity labeled for an image in the KITTI sequence</td>
</tr>
<tr>
<td>4.4</td>
<td>Experiments with the KITTI dataset: Polar representation of the disparity map after ground plane removal, $\theta$-disparity before and after labelling</td>
</tr>
<tr>
<td>4.5</td>
<td>Obstacles detection results for the KITTI dataset: overlaid segmentation results obtained in both $\theta$ and U-disparity images</td>
</tr>
<tr>
<td>4.6</td>
<td>Virtual environment: Left image; Disparity map; Ground plane mask</td>
</tr>
<tr>
<td>4.7</td>
<td>U-disparity and U-disparity labeled for an image in the virtual environment sequence</td>
</tr>
<tr>
<td>4.8</td>
<td>Virtual environment: Polar representation of the disparity map after ground plane removal, $\theta$-disparity before and after labelling</td>
</tr>
<tr>
<td>4.9</td>
<td>Obstacles detection results for the virtual environment dataset: overlaid segmentation results obtained in both $\theta$ and U-disparity images</td>
</tr>
</tbody>
</table>
| 4.10   | First row: KITTI - left and right frame, disparity map and ground truth labels
Second row: ground area for each surface from 1 to 4 obtained using method 1 3D points
Third row: ground area for each surface from 1 to 4 obtained using method 2 3D points |
| 4.11   | First row: SOV - left and right frame, disparity map and ground truth labels
Second row: ground area for each surface from 1 to 4 obtained using method 1 3D points
Third row: ground area for each surface from 1 to 4 obtained using method 2 3D points |
| 4.12   | Image based visual servoing architecture |
| 4.13   | Camera, object and robot base frames |
| 4.14   | Plant model for prediction computing |
| 4.15   | Reference trajectory |
| 4.16   | Visual Predictive Control Architecture |
| 4.17   | Visual predictive control architecture |
| 4.18   | Image moments predictor |
4.19 Visual servoing structure ........................................ 108
4.20 The visual servoing system ..................................... 110
4.21 Point features configuration for general object: (a) Initial; (b) Desired ........................................ 111
4.22 The linear and angular camera velocity for a general object: (a) image moments - Proportional law; (b) image moments - Predictive controller. ........................................ 112
4.23 The initial and desired images ..................................... 112
4.24 The linear and angular camera velocity for: a),c) image moments based proportional control law; b),d) image moments based predictive control law with reference trajectory ........................................ 113
# List of tables

<table>
<thead>
<tr>
<th>Section</th>
<th>Table Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Performance evaluation for $X^*_{alg}$ solutions</td>
<td>44</td>
</tr>
<tr>
<td>2.2</td>
<td>Performance evaluation for $X^{††}_{alg}$ solutions</td>
<td>45</td>
</tr>
<tr>
<td>4.1</td>
<td>Kitti sequence 1, 30 frames</td>
<td>92</td>
</tr>
<tr>
<td>4.2</td>
<td>Kitti sequence 2, 25 frames</td>
<td>92</td>
</tr>
<tr>
<td>4.3</td>
<td>SOV sequence, 35 frames</td>
<td>93</td>
</tr>
<tr>
<td>4.4</td>
<td>D-H Parameters</td>
<td>109</td>
</tr>
</tbody>
</table>
Chapter 1

State of the art and main contributions

From a practical engineering viewpoint, driving a robotic system implies having modular algorithms that allow any task to be decomposed into individual sub-tasks which can be independently evaluated.

My research work concerns development and implementation of motion parameterization, path panning and control techniques for robotic systems. Contributions in each of these area will be detailed and practical applications will be discussed.

**Parameterization of rigid bodies displacement and motion** represents an important research subject in multiple areas [1–5]. Rigid bodies represent basic primitives in the modeling of robotic plants [6]. A rigid body can be characterized through different types of features, among them being points and lines. Starting with classical manipulator robot kinematics and dynamics description [7] and finishing with results obtained in robotics [8, 9], machine vision [10], biomechanics [11], relative orbital motion [12, 13] or neuroscience [14], the range of applications involving points or lines transformation is very large. If points are considered then any coordinates transformation can be parametrized using homogeneous transformations [15]. For line features, parameterization techniques were developed using the dual numbers theory [16]. A parameterization performance analysis over dual orthogonal matrices, dual unit quaternions, dual special unitary matrices and dual Pauli spin matrices was conducted in [17]. The combination of dual numbers, dual vectors or dual matrices calculus with elements of screw theory generates different techniques for rigid body motion modeling [18, 19]. The desire to create a complete framework for rotation and rigid motion parameterization leads to closer inspection of linear invariants of the dual rotation matrix and the dual Euler-Rodriques parameters of rigid motion [20, 21]. Motor transformation rules and the dual inertia operator can also be used to design a general expression for the three-dimensional dynamic equation of a rigid body [22], when an arbitrary reference frame is considered.
2 State of the art and main contributions

Tensor analysis expresses the invariance of the laws of physics with respect to the change of basis and change of frame operations [23]. Orthogonal dual tensors are a complete tool for computing rigid body displacement and motion parameters. A reduced number of algebraic equations and a more compact notation with fewer variables are two of the advantages of orthogonal dual tensor based parameterization methods. Until 2014, only a few papers that analyze methods for rigid body motion parameterization using tensors were published [24, 25]. Regarding objectives, my research in this area aimed at giving a more compact algebraic description of rigid body motion using tensors and to discuss the advantages over existing methods involving dual matrices [20, 25].

Tensorial parameterization is obtained using the dyadic product between dual vectors and their reciprocals. The properties of this product generate free of coordinates solutions for different kinematic problems. The main contributions to rigid motion parameterization include: new features based rigid body pose recovery, new computational methods for the screw axis (SA) and instantaneous screw axis (ISA) motion parameters and new simultaneous closed-form solutions to the \( AX = XB \) sensor calibration problem.

Chapter 2 includes results published in multiple papers, five of them being:


Path planning procedures for autonomous robotic systems is a subject tackled by many researchers, such that a given mission is accomplished under an optimality criterion [26–30]. The missions are usually related to standard problems such as navigation, coverage, localization and mapping [31, 32]. Some works provide strategies directly implementable on particular robots with complicated dynamics and multiple sensors [33]. Other research aim to increase the task expressiveness [34], e.g. starting from Boolean inspired specifications [35] up to temporal logic ones [36, 37], even if the obtained plans may be applied only to simple robots.

It is often common to construct discrete models for the environment and robot movement capabilities, by using results from multiple areas as systems theory, computational geometry [38] and discrete event systems [39].

The contribution of my research is focused on solving some sample gathering problems. The considered task belongs to the general class of optimal assignment problems [40], since the sample can correspond to jobs and the robots to machines. The minimization of the overall time for gathering all samples (yielded by the “slowest” agent) thus translates to so-called min-max problems [41] or bottleneck assignment problems [40](ch. 6.2). However, these standard frameworks do not consider different numbers of jobs and machines, nor machines (agents) with limited energy amounts.

A broad taxonomy of allocations in multi-robot teams is presented in [42, 43], according to which the tackled problem belongs to the class of assignments of single-robot tasks (one task requiring one robot) in multi-task robot systems (a robot can move, pick-up and deposit samples). Again, the general solutions assumes utility estimates for different job-machine pairs.

For such problems, various MILP formulations are given as in [42, 44]. Some resemble to the considered problem, but they are not an exact fit because of the specificities that all samples should be eventually gathered into the same node, a robot can carry one sample at a time, there are limited amounts of energy, and we do not know a priori a relation between number of samples and number of robots. Furthermore, we provide a second MILP for the case of problems infeasible due to energy requirements. We further relax the complex MILP solutions into sub-optimal solutions as non-convex QP and iterative heuristics. The goal is to draw rules of choosing the appropriate method for a given problem, on the basis of extensive tests.

Some preliminary mathematical formulations that generalize traveling salesman problems are included in [45], with targeted application to exploring robots that have to collect and analyze multiple heterogeneous samples from a planetary surface. Other works focus on specific applications as task allocation accomplished by agents with different dynamics
[46], or allocation in scenarios with heterogenous robots that can perform different tasks [47]. Various works propose auction-based mechanisms for various assignments problems or develop and apply distributed algorithms for specific cases with equal number of agents and tasks [48]. Research [49] assumes precedence constraints on available tasks and builds solutions based on integer programming forms and auction mechanisms. However, auction-based methods are not included, the closest solution being an iterative heuristic algorithm which can be view as a specific greedy allocation method [42].

Chapter 3 includes results published in multiple papers, five of them being:


- A-E Cozma, V-G Seliman, A. Burlacu, M. Kloetzer, Numerical Simulations and Comparative Study for Auction-based Sample Gathering Solutions, 19th Int. Conf. on System Theory, Control and Computing, pp. 93-98, 2015

Perception and control of robotic systems using visual data is one of the most important research subjects in robotics and closely related areas [50–54].

Increased flexibility, accuracy, and robustness using information from images is one of the main research themes in intelligent manufacturing. Adding visual measurements to a manipulation system leads to performance improvement in multiple manufacturing areas. Visual servoing has been studied in various forms for more than three decades starting from simple pick and place tasks to today's advanced manipulation of objects. The main goal of visual servoing systems is to control the end-effector of a robot arm such as a set of image features reaches a desired configuration [55]; [56]. Visual servoing architectures can be divided into: image-based (IBVS), position-based (PBVS) and hybrid (HBS). Visibility constraints, robotic system constraints, and convergence when dealing with large displacements are problems that can appear when an IBVS architecture is used. In order to solve these problems, advanced techniques from control theory have been adapted to visual servoing systems [57], the new approach became known as Visual Predictive Control (VPC). Most of the VPC laws were developed using Nonlinear Model Predictive Control (NMPC) [58], an approach which will be detailed in this thesis also in this thesis.

Image-based control laws can be designed using different types of visual features which often are divided into: point features (centroids or corners), line or ellipse features, and generic descriptors (image moments) [59]. As known, point features based control laws have different stability problems [60] when particular configurations are considered. Also, crossing between various numbers of point features can generate discontinuities [61] which can lead to the end of the control process. Aiming to design robust, stable, and decoupled image-based control laws, a new type of visual features was proposed for servoing applications: image moments [62]. Image moments represent generic visual features that can be computed easily from a binary or a segmented image, or from a set of point features [60]. Combining image moments with VPC architectures implies extended reliability in real-time applications.

Chapter 4 includes results published in multiple papers, five of them being:

- S. Caraiman, O. Zvoristeaneu, A. Burlacu, P. Herghelegiu, Stereo Vision Based Sensory Substitution for the Visually Impaired, Sensors, vol. 19, pp. 2771-2789, 2019

- A. Burlacu, A. Baciu, V-I Manta, S. Caraiman, Ground Geometry Assessment in Complex Stereo Vision Based Applications, 21st Int. Conf. on System Theory, Control and Computing (ICSTCC), pp. 558-563, 2017


Chapter 2

Motion parameterization using dual Lie algebras

In this chapter there are presented the contributions made in the area of rigid body motion parameterization. The chapter is organized as follows: first a set of mathematical preliminaries on dual Lie algebras are detailed. Next, the techniques for rigid body motion parameterization using orthogonal dual tensors are unveiled. In the end of the chapter two applications illustrate the advantages of the proposed techniques.

2.1 Mathematical Preliminaries

Dual Numbers

Consider
\[ \mathbb{R} = \mathbb{R} + \varepsilon \mathbb{R} = \{ a = a + \varepsilon a_0 | a, a_0 \in \mathbb{R}, \varepsilon^2 = 0 \}, \]
(2.1)
to denote the set of real dual numbers, where \( a = \text{Re}(a) \) is the real part of \( a \) and \( a_0 = \text{Du}(a) \) the dual part.

The following algebraic operations with dual numbers can be defined:

- Addition "\( + \)" : \( \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R} \), \( (a, b) \rightarrow a + b \). If \( a = a + \varepsilon a_0 \), \( b = b + \varepsilon b_0 \) then:
  \[ a + b \overset{\text{def}}{=} (a + b) + \varepsilon(a_0 + b_0). \]
  (2.2)

- Multiplication with real numbers "\( \cdot \)" : \( \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R} \), \( (\lambda, a) \rightarrow \lambda a \) is defined as:
  \[ \lambda a \overset{\text{def}}{=} \lambda a + \varepsilon \lambda a_0, \forall \lambda \in \mathbb{R}, \forall a \in \mathbb{R} \]
  (2.3)
• Multiplication "·" : \( \mathbb{R} \times \mathbb{R} \to \mathbb{R} \), \((a, b) \to a \cdot b\) is defined as:

\[
a \cdot b = ab + \varepsilon (a_0 b + ab_0), \quad \forall a, b \in \mathbb{R}.
\]  

(2.4)

Using (2.2) and (2.4) results that the set of dual numbers is a unitary ring with zero divisors. The dual numbers set combined with (2.2), (2.3) is a linear space denoted by \((\mathbb{R}, +, \cdot \mathbb{R})\). Based on the property \(a \cdot (b + c) = a \cdot b + a \cdot c\), it results that \((\mathbb{R}, +, \cdot \mathbb{R})\) and (2.4) is a commutative \(\mathbb{R}\)-algebra of order 2.

Among the many properties of dual numbers, the magnitude and the inverse are the ones mostly used in this paper. The magnitude of a dual number fulfills \(|a|^2 = a^2\) and can be computed using \(|a| = |a| + \varepsilon \text{sgn}(a) a_0\), while its inverse, denoted by \(a^{-1} \in \mathbb{R}\), exists if and only if \(\text{Re}(a) \neq 0\) and is computed using \(a^{-1} = \frac{1}{a} = \frac{1}{a} - \varepsilon \frac{a_0}{a^2}\). Also, \(a \in \mathbb{R}\) is a zero divisor if and only if \(\text{Re}(a) = 0\). Based on these properties results that \((\mathbb{R}, +, \cdot \mathbb{R})\) is a commutative and unitary ring and any element \(a \in \mathbb{R}\) is either invertible or zero divisor.

Any differentiable function \(f : S \subset \mathbb{R} \to \mathbb{R}, f = f(a)\) can be completely defined on \(S \subset \mathbb{R}\) such that:

\[
f : S \subset \mathbb{R} \to \mathbb{R}; \quad f(a) = f(a) + \varepsilon a_0 f'(a).
\]  

(2.5)

Based on the previous property, two of the most important functions have the following expressions: \(\cos a = \cos a - \varepsilon a_0 \sin a; \sin a = \sin a + \varepsilon a_0 \cos a\).

**Remark 1.** For any \(a, b \in \mathbb{R}, a = a + \varepsilon a_0; b = b + \varepsilon b_0\) with \(\text{Re}(a^2 + b^2) \neq 0\), the \(\text{atan2}\) function gives the dual number:

\[
\text{atan2}(b, a) = \text{atan2}(b, a) + \varepsilon \frac{b_0 a - ba_0}{a^2 + b^2}.
\]  

(2.6)

**Proof.** Let \(\text{atan2}(b, a) = \alpha\) and \(\sin \alpha = \frac{b}{\sqrt{a^2 + b^2}}\), \(\cos \alpha = \frac{a}{\sqrt{a^2 + b^2}}\). Taking into account that \(a = a + \varepsilon a_0, b = b + \varepsilon b_0\) implies \(a^2 + b^2 = a^2 + b^2 + \varepsilon (2a a_0 + 2b b_0)\), based on (2.5) results that:

\[
\frac{b}{\sqrt{a^2 + b^2}} = \frac{b}{\sqrt{a^2 + b^2}} + \varepsilon \frac{1}{\sqrt{a^2 + b^2}} \left( \frac{b_0 a^2 - b a_0}{a^2 + b^2} \right)
\]

\[
\frac{a}{\sqrt{a^2 + b^2}} = \frac{a}{\sqrt{a^2 + b^2}} + \varepsilon \frac{1}{\sqrt{a^2 + b^2}} \left( \frac{a_0 b^2 - b b_0}{a^2 + b^2} \right).
\]  

(2.7)
2.1 Mathematical Preliminaries

The structure of the dual angle $\alpha = \alpha + \varepsilon d$ yields $\sin \alpha = \sin \alpha + \varepsilon d \cos \alpha$; $\cos \alpha = \cos \alpha - \varepsilon d \cos \alpha$, and after some direct algebraic calculus, the followings can be identified

$$\alpha = \text{atan}2(b, a)$$
$$d = \frac{b_0a - ba_0}{a^2 + b^2}, a \neq 0 \text{ and } b \neq 0$$

(2.8)

which proves the remark.

A direct result of the previous Remark is related to the multiplication of $a$ and $b$ with a dual number $k$ ($\text{Re}(k) \neq 0$) which implies that $\text{atan}2(kb, ka) = \text{atan}2(b, a)$.

**Dual vectors**

The linear space of free Euclidean vectors with dimension 3 will be denoted by $V_3$. The ensemble of dual vectors is defined as

$$\overline{V_3} = V_3 + \varepsilon V_3 = \{a = a + \varepsilon a_0; a, a_0 \in V_3, \varepsilon^2 = 0\},$$

(2.9)

where $a = \text{Re}(a)$ is the real part of $a$ and $a_0 = \text{Du}(a)$ the dual part.

The following algebraic operations can be defined for dual vectors:

- Addition $" + " : \overline{V_3} \times \overline{V_3} \to \overline{V_3}$: $(\lambda, a) \to \lambda a$, where $\lambda = \lambda + \varepsilon \lambda_0$, $a = a + \varepsilon a_0$, is defined as:

$$\lambda a = \lambda a + \varepsilon (\lambda_0 a + \lambda a_0).$$

(2.11)

The dual vectors set combined with (2.10) forms a commutative group. Using the properties

$$\lambda(a + b) = \lambda a + \lambda b, \forall \lambda \in \mathbb{R}, \forall a, b \in V_3$$

$$(\lambda + \mu)a = \lambda a + \mu a, \forall \lambda, \mu \in \mathbb{R}, \forall a \in V_3$$

$$\lambda(\mu a) = (\lambda \mu)a, \forall \lambda, \mu \in \mathbb{R}, \forall a \in V_3$$

$$1 \cdot a = a, \forall a \in V_3$$

results that $(V_3, +, \cdot, \mathbb{R})$ is a module over the unitary ring $\mathbb{R}$, also called $\mathbb{R}$-module [63].
In $V_3$ three products for dual vectors can be defined. First, the scalar product ’·’: $V_3 \times V_3 \rightarrow \mathbb{R}$:
\[
(a, b) \rightarrow a \cdot b = a \cdot b + \varepsilon (a_0 \cdot b + a \cdot b_0).
\] (2.12)
From (2.12) it results that a unit dual vector is a dual vector $u \in V_3$ with $u^2 = u \cdot u = 1$. The second product is the cross product ’×’: $V_3 \times V_3 \rightarrow V_3$
\[
a \times b \overset{def}{=} a \times b + \varepsilon (a_0 \times b + a \times b_0).
\] (2.13)
The last product, scalar triple, is a combination between (2.12) and (2.13):
\[
< a, b, c > = a \cdot (b \times c).
\] (2.14)
If $\text{Re}(< a, b, c >) \neq 0$ then the three dual vectors are linear independent. Otherwise, if from $\alpha a + \beta b + \gamma c = 0$, in which at least one of $\{ \alpha, \beta, \gamma \}$ has the real part $\neq 0$, implies that $< a, b, c >= 0$ then $\{ a, b, c \}$ are linear dependent.

The magnitude of a dual vector $a$, denoted by $|a|$, is the dual number which fulfills $|a| \cdot |a| = a \cdot a$ and can be computed using
\[
|a| = \begin{cases} 
\|a\| + \varepsilon \frac{a_0 \cdot a}{\|a\|}, & \text{Re}(a) \neq 0 \\
\varepsilon \|a_0\|, & \text{Re}(a) = 0
\end{cases},
\] (2.15)
where $\|.|\|$ is the Euclidean norm. For any dual vector $a \in V_3$, if $|a| = 1$ then $a$ is called unit dual vector.

**Theorem 1.** For any $a \in V_3$, a dual number $\alpha \in \mathbb{R}$, and a unit dual vector $u_a \in V_3$ exist in order to have:
\[
a = \alpha \ u_a.
\] (2.16)
The computational formulas for $\alpha$ and $u_a$ are:
\[
\pm \alpha = |a|,
\] (2.17)
\[
\pm u_a = \begin{cases} 
\frac{a}{\|a\|} + \varepsilon \frac{a_0 \times a}{\|a\|^3}, & \text{Re}(a) \neq 0 \\
\frac{a_0}{\|a_0\|} + \varepsilon v \times \frac{a_0}{\|a_0\|}, & \forall v \in V_3, \text{Re}(a) = 0
\end{cases}.
\] (2.18)
Also, for $\text{Re}(a) \neq 0$, $\alpha$ and $u_a$ are unique up to a sign change.
The proof of this Theorem was presented by the authors in [5]. Theorem 1 underlines that any dual vector \( \mathbf{a} \in \mathbb{V}_3 \), with \( \text{Re}(\mathbf{a}) \neq 0 \) can be associated with a labeled directed line in the Euclidean three dimensional space (Fig. 1). The elements of the unit dual vector \( \mathbf{u}_a = \mathbf{u} + \epsilon \mathbf{u}_0 \) describe a line parametrized by Plucker coordinates, while the dual number \( \alpha = |\mathbf{a}| = \frac{||\mathbf{a}|| + \epsilon \mathbf{a}_0 \cdot \mathbf{a}}||\mathbf{a}|| \) represents the label. Regarding parametric representation, for \( \text{Re}(\mathbf{a}) \neq 0 \) the equation is \( \mathbf{r} = \mathbf{a} \times \mathbf{a}_0 = \frac{\mathbf{a} \times \mathbf{a}_0}{||\mathbf{a}||^2} + \lambda \frac{\mathbf{a}}{||\mathbf{a}||}, \forall \lambda \in \mathbb{R} \). The case \( \text{Re}(\mathbf{a}) = 0 \) will prove itself very important for future results. For this case the geometrical representation is a set of parallel lines described by \( \mathbf{a}_0 = \frac{\mathbf{a}_0}{||\mathbf{a}_0||} \) and labeled with \( \alpha = |\mathbf{a}| = \epsilon ||\mathbf{a}_0|| \). Also, the parametric equation is \( \mathbf{r} = \mathbf{v} + \lambda \frac{\mathbf{a}_0}{||\mathbf{a}_0||}, \forall \mathbf{v} \in \mathbb{V}_3, \forall \lambda \in \mathbb{R} \).

**Dual Tensors**

An \( \mathbb{R} \)-linear application of \( \mathbb{V}_3 \) into \( \mathbb{V}_3 \) is called an Euclidean dual tensor:

\[
\begin{align*}
T(\lambda_1 \mathbf{v}_1 + \lambda_2 \mathbf{v}_2) &= \lambda_1 T(\mathbf{v}_1) + \lambda_2 T(\mathbf{v}_2), \\
\forall \lambda_1, \lambda_2 &\in \mathbb{R}, \forall \mathbf{v}_1, \mathbf{v}_2 \in \mathbb{V}_3.
\end{align*}
\]

(2.19)

From now on, any Euclidean dual tensor will be shortly called dual tensor and \( \mathbb{L}(\mathbb{V}_3, \mathbb{V}_3) \) will denote the free \( \mathbb{R} \)-module of dual tensors. Any dual tensor \( T \in \mathbb{L}(\mathbb{V}_3, \mathbb{V}_3) \) can be decomposed in \( T = T^T + \epsilon T_0 \), where \( T, T_0 \in \mathbb{L}(\mathbb{V}_3, \mathbb{V}_3) \) are real tensors. The transposed dual tensor, denoted by \( T^T \), is defined by

\[
\mathbf{v}_1 \cdot (T^T \mathbf{v}_2) = \mathbf{v}_2 \cdot (T^T \mathbf{v}_1), \forall \mathbf{v}_1, \mathbf{v}_2 \in \mathbb{V}_3,
\]

(2.20)

while the determinant is:

\[
< T \mathbf{v}_1, T \mathbf{v}_2, T \mathbf{v}_3 > = \det T < \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 >, \forall \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \in \mathbb{V}_3.
\]

(2.21)

A dual tensor \( T \in L(\mathbb{V}_3, \mathbb{V}_3) \) can be classified as being: symmetric if \( T = T^T \), skew-symmetric \( T = -T^T \), invertible if \( \text{Re}(\det T) \neq 0 \). The inverse is computed using [21] \( T^{-1} = T^{-1} + \epsilon T^{-1} T_0 T^{-1} \) and fulfills \( T T^{-1} = T^{-1} T = I \), where \( I \) is the unity orthogonal dual tensor.

For any dual vector \( \mathbf{a} \in \mathbb{V}_3 \) the associated skew-symmetric dual tensor will be denoted by \( \tilde{\mathbf{a}} \) and will be defined by:

\[
\tilde{\mathbf{a}} \mathbf{b} = \mathbf{a} \times \mathbf{b}, \forall \mathbf{b} \in \mathbb{V}_3.
\]

(2.22)
For any skew-symmetric dual tensor a unique defined dual vector $a = vect A, a \in V_3$ exists in order to have $A b = a \times b, \forall b \in V_3$. The set of skew-symmetric dual tensors is structured as a free $R$-module of rank 3, and is isomorphic with $V_3^3$ [5].

An important class of invariants for a dual tensor are the linear invariants and are denoted by [5, 64]

\[
\text{vect} T = \text{vect} \frac{1}{2} [T - T^T] \\
\text{trace} T = \frac{<Tv_1,v_2,v_3> + <v_1Tv_2,v_3> + <v_1,v_2Tv_3>}{<v_1,v_2,v_3>},
\]

for any $v_1, v_2, v_3 \in V_3$ with $Re(<v_1,v_2,v_3>) \neq 0$.

Consider $v$ to be a generic dual vector. If $B = \{e_1, e_2, e_3\}$ is a dual basis then $v = v^k e_k$. The components $v^k$ can be stacked as a matrix with one column

\[
[v] = \begin{bmatrix} v^1 \\ v^2 \\ v^3 \end{bmatrix}.
\]

For any dual tensor $T = T^i_j e_i \otimes e^j$ the decomposition (2.38) leads to the following matrix:

\[
[T] = \begin{bmatrix} T^1_1 & T^1_2 & T^1_3 \\ T^2_1 & T^2_2 & T^2_3 \\ T^3_1 & T^3_2 & T^3_3 \end{bmatrix}.
\]

If a dual tensor is applied to a dual vector, $u = Ty$, the matrix of the resulting dual vector can be computed by: $[T] \cdot [v]$. If two dual tensors $T$ and $K$ are composed then the attached matrix of the resulting dual tensor is recovered from: $[T] \cdot [K]$.

For $a = a^i e_i$ and $b = b^j e_j$, the dyadic product $a \otimes b$ is characterized by a matrix which can be computed from

\[
[a \otimes b] = [a] G [b]^T,
\]

where $G = (g_{ij}), g_{ij} = e_i \cdot e_j$. If $B$ is orthonormal then $G = I$ and $[a \otimes b] = [a] [b]^T$. 

2.2 Rigid body motion parameterization using orthogonal dual tensors

2.2.1 Dual Tensor Construction using Dual Vectors Bases

The rigid body displacement and motion parameterization methods presented next are based on the properties of dual tensors. The key of the chosen design is the combination between dual bases and the dyadic product of dual vectors. For a proper comprehension of the proposed technique, first some algebraic results for dual bases are uncovered.

**Theorem 2.** If \( \mathbf{a} \in \mathbb{V}_3 \) then a dual number \( \lambda \in \mathbb{R} \) and a unit dual vector \( \mathbf{u} \in \mathbb{V}_3 \) exist in order to have \( \mathbf{a} = \lambda \mathbf{u} \). Also, if \( \text{Re}(\mathbf{a}) \neq 0 \) then \( \lambda \) and \( \mathbf{u} \) are unique up to a sign change.

**Proof.** If \( ||\cdot|| \) denotes the Euclidean norm then

\[
\pm \lambda = |\mathbf{a}| \quad \text{and} \quad \pm \mathbf{u} = \begin{cases} \frac{\mathbf{a}}{|\mathbf{a}|} + \varepsilon \frac{\mathbf{a} \times (\mathbf{a}_0 \times \mathbf{a})}{|\mathbf{a}|^3} & \text{Re}(\mathbf{a}) \neq 0 \\ \frac{\mathbf{a}}{|\mathbf{a}_0||\mathbf{a}|} + \varepsilon \mathbf{v} \times \frac{\mathbf{a}}{|\mathbf{a}_0||\mathbf{a}|} & \forall \mathbf{v} \in \mathbb{V}_3 \end{cases} \quad \text{Re}(\mathbf{a}) = 0.
\]

The previous result allows to geometrically describe any dual vector from the 3D Euclidean space (Figure 2.1). For a dual vector \( \mathbf{a} \) from \( \mathbb{V}_3 \), with \( \text{Re}(\mathbf{a}) \neq 0 \), a labeled line in the Euclidean three dimensional space can be associated. The elements of the unit dual vector \( \mathbf{u} = \mathbf{u} + \varepsilon \mathbf{u}_0 \) give a line parametrized by Plucker coordinates [33], while the dual number \( \lambda = |\mathbf{a}| = |\mathbf{a}| + \varepsilon \mathbf{a}_0 \cdot \mathbf{a} \) represents the label. If \( \text{Re}(\mathbf{a}) = 0 \) then the geometrical interpretation is a set of parallel lines described by \( \frac{\mathbf{a}_0}{|\mathbf{a}_0||\mathbf{a}|} \) and labeled with \( \lambda = |\mathbf{a}| = \varepsilon |\mathbf{a}_0||\mathbf{a}| \).

**Definition 1.** A set of three dual vectors \( \mathbf{B} = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\} \) will be called dual basis if the dual vectors are \( \mathbb{R} \) linear independent and also represent a span set for \( \mathbb{V}_3 \).

**Proposition 1.** Any \( \mathbf{e}_k \in \mathbb{V}_3, k = 1, 3 \) that fulfill \( \text{Re}(<\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3>) \neq 0 \) are \( \mathbb{R} \) linear independent.

**Proof.** Consider the dual numbers \( \alpha_i = \alpha_i + \varepsilon \alpha_{i0}, i = 1, 3 \) which can lead to

\[
\alpha_1 \mathbf{e}_1 + \alpha_2 \mathbf{e}_2 + \alpha_3 \mathbf{e}_3 = 0.
\]

Based on the definition of dual numbers results that

\[
\alpha_1 \mathbf{e}_1 + \alpha_2 \mathbf{e}_2 + \alpha_3 \mathbf{e}_3 = 0 \quad (2.29)
\]
Motion parameterization using dual Lie algebras

Fig. 2.1 Geometrical interpretation of Theorem 1

and

$$\sum_{i=1}^{3} (\alpha_i e_{i0} + \alpha_{0i} e_i) = 0.$$  \hspace{1cm} (2.30)

From \(\text{Re}(\langle e_1, e_2, e_3 \rangle) \neq 0\) results that \(\langle e_1, e_2, e_3 \rangle \neq 0\), which combined with (2.29) make \(\alpha_1 = \alpha_2 = \alpha_3 = 0\). Further, adding (2.30) generates \(\sum_{i=1}^{3} \alpha_{0i} e_i = 0\). The previous result combined with the fact that \(\langle e_1, e_2, e_3 \rangle \neq 0\), show that \(\alpha_{10} = \alpha_{02} = \alpha_{30} = 0\), which proves the proposition.

**Proposition 2.** If any three dual vectors \(e_k \in V_3\), \(k = 1, 3\), fulfill \(\text{Re}(\langle e_1, e_2, e_3 \rangle) \neq 0\) then there are uniquely determined \(\{e^1, e^2, e^3\}\) using the conditions \(e^i \cdot e^j = \delta^j_i, i, j = 1, 3\), where \(\delta^j_i\) is the Kronecker symbol.

**Proof.** Let \(\{e_1, e_2, e_3\}\) be a set constructed by the following rules:

$$e^1 = \frac{e_2 \times e_3}{\langle e_1, e_2, e_3 \rangle}, \quad e^2 = \frac{e_3 \times e_1}{\langle e_1, e_2, e_3 \rangle}, \quad e^3 = \frac{e_1 \times e_2}{\langle e_1, e_2, e_3 \rangle}.$$  \hspace{1cm} (2.31)

Using (2.31) the conditions \(e^i \cdot e^j = \delta^j_i, i, j = 1, 3\) are fulfilled.

**Theorem 3.** The ensemble \(V_3\) together with the sum of dual vectors and the multiplication of dual vectors with dual numbers, is a free \(\mathbb{R}\)-module and its rank is 3.

**Proof.** Let \(\{e_1, e_2, e_3\}\) be three dual vectors that fulfill \(\text{Re}(\langle e_1, e_2, e_3 \rangle) \neq 0\). For any \(v \in V_3\) there are uniquely determined \(\alpha^i \in \mathbb{R}, i = 1, 3\) as so \(v = \alpha^i \cdot e_i\). Previous the Einstein’s rule for mute indexes summation was considered. The values of \(\alpha^i, i = 1, 3\) are computed using
2.2 Rigid body motion parameterization using orthogonal dual tensors

\( \alpha^i = \mathbf{v} \cdot \mathbf{e}^i \). Thus it results that \((V_3,+,\cdot)\) is a free \( \mathbb{R} \)-module (Appendix B) with rank 3, while the set \( \mathcal{B} = \{ \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \} \) is a dual basis in this module.

**Remark 2.** For a dual basis \( \mathcal{B} = \{ \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \} \), the set \( \mathcal{B}^* = \{ \mathbf{e}^1, \mathbf{e}^2, \mathbf{e}^3 \} \) represents its reciprocal dual basis. The dual basis \( \mathcal{B} \) coincides with \( \mathcal{B}^* \) if and only if \( \mathcal{B} \) is an orthonormal basis (aka \( \mathbf{e}_j \cdot \mathbf{e}_j = \delta_{jj} \)).

**Definition 2.** A basis \( \mathcal{B} = \{ \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \} \) from \( V_3^\mathbb{R} \) will be referred to as a dual rigid basis if

\[
\begin{align*}
\mathbf{e}_j \cdot \mathbf{e}_j &= \mathbf{e}_0^i \cdot \mathbf{e}_0^j, \quad i, j = 1, 2, 3, \\
\langle \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \rangle &= \langle \mathbf{e}_0^1, \mathbf{e}_0^2, \mathbf{e}_0^3 \rangle,
\end{align*}
\]

(2.32)

where \( \mathbf{e}_0^i = \mathbf{e}_i(t_0) \).

**Remark 3.** If \( \mathcal{B} = \{ \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \} \) is a dual rigid basis then \( \mathcal{B}^* = \{ \mathbf{e}^1, \mathbf{e}^2, \mathbf{e}^3 \} \), its reciprocal dual basis, is also rigid.

Given two dual vectors \( \mathbf{a} \) and \( \mathbf{b} \in V_3 \), \( \mathbf{a} \otimes \mathbf{b} \) denotes a dual tensor called **tensor (dyadic) product** and is defined by:

\[
\mathbf{a} \otimes \mathbf{b} : V_3 \times V_3 \to V_3, \quad (\mathbf{a} \otimes \mathbf{b}) \mathbf{v} = (\mathbf{v} \cdot \mathbf{b}) \mathbf{a}, \quad \forall \mathbf{v} \in V_3.
\]

(2.33)

An important property of (2.74) is that \( (\mathbf{a} \otimes \mathbf{b})(\mathbf{c} \otimes \mathbf{d}) = (\mathbf{b} \cdot \mathbf{c})\mathbf{a} \otimes \mathbf{d} \).

**Theorem 4.** The following statements are true:

1. A dual tensor \( T : V_3 \to V_3 \) is uniquely determined by the values obtained after \( T \) is applied to the elements of the dual basis \( \mathcal{B} = \{ \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \} \):

\[
T = (Te_j) \otimes \mathbf{e}_j.
\]

(2.34)

2. The ensemble \( \mathbf{L}(V_3, V_3) \) is a free \( \mathbb{R} \)-module of rank equal to 9.

**Proof.** Starting with (2.34) the Einstein’s rule for mute indexes summation, when \( i \) varies from 1 to 3, will be used. Let \( \mathbf{v} \in V_3 \) be an arbitrary vector that has the following expression in the basis \( \mathcal{B} = \{ \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \} \):

\[
\mathbf{v} = (\mathbf{v} \cdot \mathbf{e}^i) \mathbf{e}_j.
\]

(2.35)

Using (2.19) and (2.74) it results that

\[
Tv = T[(\mathbf{v} \cdot \mathbf{e}^i) \mathbf{e}_j] = (\mathbf{v} \cdot \mathbf{e}^i)(Te_j) =
\]

\[
[(Te_j) \otimes \mathbf{e}^i] \mathbf{v},
\]

(2.36)
Motion parameterization using dual Lie algebras

which proves the first part of the theorem.

If $T$ is a dual tensor then the dual vectors $T\mathbf{e}_j, j = 1, 3$ can be written as

$$T\mathbf{e}_j = [\mathbf{e}^i \cdot (T\mathbf{e}_j)] \mathbf{e}_i, i = 1, 3.$$  \hspace{1cm} (2.37)

Denoting with $T^i_j = \mathbf{e}^i \cdot (T\mathbf{e}_j)$, $T^i_j \in \mathbb{R}$ and combining (2.37) with (2.34) generates

$$T = T^i_j \mathbf{e}_i \otimes \mathbf{e}^j,$$  \hspace{1cm} (2.38)

which represents a linear combination of tensors $\{\mathbf{e}_i \otimes \mathbf{e}^j\}_{i,j=1,3}$ that is equivalent with a spanning set. The previous result, together with the remark (which can be easily proven) that $\{\mathbf{e}_i \otimes \mathbf{e}^j\}_{i,j=1,3}$ are $\mathbb{R}$ linearly independent in $L$, imply that $\{\mathbf{e}_i \otimes \mathbf{e}^j\}_{i,j=1,3}$ is a basis in $L(V_3, V_3)$ and $rank_{\mathbb{R}} L(V_3, V_3) = 9$.

Any dual tensor $T$ can be characterized by three dual numbers which are its principal invariants $(T_I, T_{II}, T_{III})$. The principal invariants can be computed using:

$$Tv = \lambda \mathbf{v}, \lambda \in \mathbb{R}, \mathbf{v} \in V_3, Re(\mathbf{v}) \neq 0.$$  \hspace{1cm} (2.39)

From (2.39) it derives that

$$[T - \lambda I] \mathbf{v} = 0,$$  \hspace{1cm} (2.40)

where $I$ is the identity dual tensor. Because $I = \mathbf{e}_i \otimes \mathbf{e}^i$ it follows that:

$$\{[T\mathbf{e}_i - \lambda \mathbf{e}_i] \otimes \mathbf{e}^i\} \mathbf{v} = 0.$$  \hspace{1cm} (2.41)

Applying (2.74) to (16) leads to:

$$(\mathbf{v} \cdot \mathbf{e}^i)[T\mathbf{e}_i - \lambda \mathbf{e}_i] = 0, Re(\mathbf{v}) \neq 0.$$  \hspace{1cm} (2.42)

This result emerges from the fact that when $Re(\mathbf{v}) \neq 0$, at least one of the scalars $\mathbf{v} \cdot \mathbf{e}^i, i = 1, 3$ is not zero, and it can be concluded that vectors $[T\mathbf{e}_i - \lambda \mathbf{e}_i], i = 1, 3$ are linearly dependent. Therefore, their scalar triple product is null:

$$< T\mathbf{e}_1 - \lambda \mathbf{e}_1, T\mathbf{e}_2 - \lambda \mathbf{e}_2, T\mathbf{e}_3 - \lambda \mathbf{e}_3 >= 0.$$  \hspace{1cm} (2.43)

Based on the properties of the scalar triple product the characteristic equation of $T$ is

$$\lambda^3 - T_I \lambda^2 + T_{II} \lambda - T_{III} = 0.$$  \hspace{1cm} (2.44)
2.2 Rigid body motion parameterization using orthogonal dual tensors

where the dual numbers \( T_I, T_{II}, T_{III} \) are:

\[
T_I = \frac{<T e_1, e_2, e_3> + <e_1, T e_2, e_3> + <e_1, e_2, T e_3>}{<e_1, e_2, e_3>}, \tag{2.45}
\]

\[
T_{II} = \frac{<e_1, T e_2, T e_3> + <T e_1, e_2, e_3> + <T e_1, T e_2, e_3>}{<e_1, e_2, e_3>}, \tag{2.46}
\]

\[
T_{III} = \frac{<T e_1, T e_2, T e_3>}{<e_1, e_2, e_3>}. \tag{2.47}
\]

The dual scalars \( T_I, T_{II}, T_{III} \) do not depend on how the basis \( B = \{ e_1, e_2, e_3 \} \) is chosen, thus making them suitable to characterize the dual tensor \( T \). The first scalar invariant is usually denoted \( T_I = \text{trace} T \) and the third one by \( T_{III} = \det T \). Based on the Cayley-Hamilton’s theorem, which states that any tensor verifies its characteristic equation, results that

\[
T^3 - T_I T^2 + T_{II} T - T_{III} I = O, \tag{2.48}
\]

where \( O \) is the null tensor.

For any dual vector \( \bar{a} \in V_3 \), the associated skew-symmetric dual tensor will be denoted by \( \tilde{a} \) and defined by:

\[
\tilde{a} b = a \times b, \forall b \in V_3. \tag{2.49}
\]

The set of skew-symmetric dual tensors is structured as a free \( \mathbb{R} \)-module of rank 3, module which is isomorph with \( V_3 \). The following notation are considered \( a = \text{vect} \tilde{a}, \tilde{a} = \text{spin} a \).

For an arbitrary dual tensor \( T \) the following entities can be computed

\[
\text{sym} T = \frac{1}{2} [T + T^T], \quad \text{skew} T = \frac{1}{2} [T - T^T], \tag{2.50}
\]

where ”sym” is the symmetric part of the dual tensor and ”skew” is its skew-symmetric part. Also, the axial dual vector of tensor \( T \) is given by:

\[
\text{vect} T = \text{vect} \frac{1}{2} [T - T^T]. \tag{2.51}
\]

Both vect\( T \) and trace\( T \) have the \( \mathbb{R} \)-linearity property: \( \forall \lambda_1, \lambda_2 \in \mathbb{R}, \forall T_1, T_2 \in L(V_3, V_3) \)

\[
\text{vect}(\lambda_1 T_1 + \lambda_2 T_2) = \lambda_1 \text{vect} T_1 + \lambda_2 \text{vect} T_2,
\]

\[
\text{trace}(\lambda_1 T_1 + \lambda_2 T_2) = \lambda_1 \text{trace} T_1 + \lambda_2 \text{trace} T_2. \tag{2.52}
\]

The previous information is a second class of invariants that can be used to describe the dual tensor, and are called linear invariants [64]. Considering the skew-symmetric dual tensor
\[ \tilde{a}, \text{ its principal invariants } \tilde{a}_I \text{ and } \tilde{a}_{II} \text{ are equal to } 0, \text{ while } \tilde{a}_{II} = |a|^2. \text{ Thus, equation (2.48) leads to} \]
\[ \tilde{a}^3 = -|a|^2 \tilde{a}. \]  

(2.53)

The linear invariants are \( \text{vect } \tilde{a} = a \) and \( \text{trace } \tilde{a} = 0 \). If \( u \) is the unit dual vector then \( u^3 = -\tilde{u} \).

If the dual tensor defined by (2.74) is analyzed, the following results emerge: \( (a \otimes b)^T = b \otimes a, \text{ vect}(a \otimes b) = \frac{1}{2}(b \times a) \) and \( \text{trace}(a \otimes b) = a \cdot b \). These results combined with (2.52), when \( T \) is given by (2.34), lead to:

\[ T^T = e^i \otimes (T e^i), \]  

(2.54)

\[ \text{vect} T = \frac{1}{2}(T e^i) \times e^i, \]  

(2.55)

\[ \text{trace} T = (T e^i) \cdot e^i. \]  

(2.56)

In (2.54), (2.55), (2.56) the Einstein’s rule for mute indexes summation has been used, where \( i \) varies from 1 to 3.

**Orthogonal Dual Tensors**

Consider \( SO_3 \) to be the set of real special orthogonal tensors [64]. Let the orthogonal dual tensor set be denoted by

\[ SO_3 = \{ R \in L(V_3, V_3) \mid RR^T = I, \det R = 1 \}, \]  

(2.57)

where \( I \) is the unit orthogonal dual tensor. The internal structure of any orthogonal dual tensor \( R \in SO_3 \) is illustrated in a series of results detailed in [5]:

**Theorem 5** (Structure Theorem). For any \( R \in SO_3 \), a unique decomposition is viable [5]

\[ R = (I + \varepsilon \hat{\rho})Q, \]  

(2.58)

where \( Q \in SO_3 \) and \( \rho = \text{vect} \hat{\rho}, \rho \in V_3 \) are called **structural invariants**.

Taking into account the Lie group structure of \( SO_3 \) [65] and the result presented in the previous theorem, it can be concluded that any orthogonal dual tensor \( R \in SO_3 \) can be used to globally parameterize displacements of rigid bodies.

**Theorem 6** (Representation Theorem). For any orthogonal dual tensor \( R \) defined as in (2.58), a dual number \( \alpha = \alpha + \varepsilon d \) and a dual unit vector \( u = u + \varepsilon u_0 \) can be computed in order to
have [5]:

\[ R(\alpha, \mathbf{u}) = I + \sin \alpha \mathbf{u} + (1 - \cos \alpha)\mathbf{u}^2 = \exp(\alpha \mathbf{u}). \]  

(2.59)

The computational formulas for \( \alpha, \mathbf{u}, d, \mathbf{u}_0 \), are:

\[ \alpha = \text{atan2}(\pm \frac{1}{2} \sqrt{(1 + \text{trace}Q)(3 - \text{trace}Q)}, \frac{\text{trace}Q - 1}{2}). \]

\[ \mathbf{u} = \begin{cases} 
\pm \text{vect} \left( \frac{1}{\sqrt{(1 + \text{trace}Q)(3 - \text{trace}Q)}}(Q - Q^T) \right), \\
\text{when } \text{trace}Q \in (-1, 3); \\
\frac{Q\mathbf{v} + \mathbf{v}}{||Q\mathbf{v} + \mathbf{v}||}, \forall \mathbf{v} \in V_3, \\
\text{when } \text{trace}Q = -1 (Q \text{ is symmetric}); \\
\frac{\rho}{||\rho||}, \\
\text{when } \text{trace}Q = 3 (Q = I); 
\end{cases} \]  

(2.60)

\[ d = \rho \cdot \mathbf{u} \]

\[ \mathbf{u}_0 = \begin{cases} 
\frac{1}{2} \rho \times \mathbf{u} + \frac{1}{2} \cot \frac{\alpha}{2} \mathbf{u} \times (\rho \times \mathbf{u}) & \alpha \neq 0; \\
\frac{1}{2} \rho \times \mathbf{u} & \alpha = 0; 
\end{cases} \]

The parameters \( \alpha \) and \( \mathbf{u} \) are called the natural invariants of \( R \in SO_3 \). The unit dual vector \( \mathbf{u} \) gives the Plucker representation of the Mozzi-Chalses axis [21], while the dual angle \( \alpha = \alpha + \varepsilon d \) contains the rotation angle \( \alpha \) and the translation distance \( d \).

**Theorem 7.** The recovery of \( \alpha, \mathbf{u} \) can also be done using the linear and structural invariants. This leads to:

\[ \alpha = \text{atan2}(\pm ||\text{vect}R||, \frac{1}{2} [\text{trace}R - 1]). \]  

(2.61)
\[ u = \begin{cases} \pm \frac{\text{vect}R}{|\text{vect}R|}, & \text{when } \text{Re}(\text{vect}R) \neq 0; \\ \frac{Qv + v}{||Qv + v||} + \frac{1}{2} \rho \times \frac{Qv + v}{||Qv + v||}, & \forall v \in V_3 \\ \rho ||\rho||, & \text{when } \text{Re}(\text{vect}R) = 0 \text{ and } \text{trace}Q = -1; \\ \rho ||\rho||, & \text{when } \text{Re}(\text{vect}R) = 0 \text{ and } \text{trace}Q = 3; \end{cases} \] (2.62)

**Proof.** The proof is a result of

\[ u \sin \alpha = \text{vect}R, \]
\[ \cos \alpha = \frac{1}{2} [\text{trace}R - 1], \] (2.63)
equations that emerge from (2.59). For more details see [5]. \qed

**Remark 4.** If \( R \) is an orthogonal dual tensor then for any dual vectors \( a \) and \( b \) the following two expression are valid:

\[ R(a) \cdot R(b) = a \cdot b, \] (2.64)
\[ R(a) \times R(b) = R(a \times b). \] (2.65)

Consider next two dual orthogonal tensors \( R(\alpha_1, u_1), R(\alpha_2, u_2) \). Combining the previous remark with the identity

\[ \tilde{Ra} = R\tilde{a}R^T \] (2.66)
the following semi-commutativity property is obtained:

**Remark 5.** For any dual angles \( \alpha_1, \alpha_2 \in \mathbb{R} \) and any dual unit vectors \( u_1, u_2 \in V_3 \), the following semi-commutativity property is valid

\[ R(\alpha_2, u_2)R(\alpha_1, u_1) = R(\alpha_1, u_1^*)R(\alpha_2, u_2), \] (2.67)

where \( u_1^* = R(\alpha_2, u_2)u_1 \).

The Lie algebra of \( \mathbb{S}^3 \) is the skew-symmetric dual tensors set denoted by \( \mathfrak{s}_3 = \{ \tilde{a} \in L(V_3, V_3) \mid \tilde{a} = -\tilde{a}^T \} \) [66], where the internal mapping is \( \langle \tilde{a}_1, \tilde{a}_2 \rangle = \tilde{a}_1 \tilde{a}_2 \).

The link between the Lie algebra \( \mathfrak{s}_3 \), the Lie group \( \mathbb{S}^3 \) and the exponential map is given by
2.2 Rigid body motion parameterization using orthogonal dual tensors

Theorem 8. The mapping

$$\exp : so_3 \rightarrow SO_3, \exp(\overline{\alpha}) = e^{\overline{\alpha}} = \sum_{k=0}^{\infty} \frac{\alpha^k}{k!},$$

is well defined and surjective.

Proof. The proof of this Theorem was presented by the authors in reference [5].

Any screw axis can be parameterized by a unit dual vector, while the screw parameters (angle of rotation about the screw and the translation along the screw axis) can be combined in a dual angle. The computation of the screw axis is linked with the problem of finding the logarithm of an orthogonal dual tensor $R$, a multifunction defined by

$$\log : SO_3 \rightarrow so_3, \log R = \left\{ \overline{\psi} \in so_3 \mid \exp(\overline{\psi}) = R \right\},$$

and is the inverse of (2.68).

Based on theorems 3 and 4, for any orthogonal dual tensor $R$ a dual vector $\overline{\psi} = \alpha \overline{u} = \overline{\psi} + \varepsilon \overline{\psi}_0$ can be computed and represents the screw dual vector which embeds the screw axis and screw parameters. The form of $\overline{\psi}$ implies that $\overline{\psi} \in log R$.

The types of rigid displacements that can be parameterized by $\overline{\psi}$ are:

(i) rototranslation if $|\overline{\psi}| \in \mathbb{R} - \varepsilon \mathbb{R}$ ($\overline{\psi} \neq 0$, $\overline{\psi}_0 \neq 0$ and $\overline{\psi} \cdot \overline{\psi}_0 \neq 0$);

(ii) pure translation if $|\overline{\psi}| \in \varepsilon \mathbb{R}$ ($\overline{\psi} = 0$ and $\overline{\psi}_0 \neq 0$);

(iii) pure rotation if $|\overline{\psi}| \in \mathbb{R}$ ($\overline{\psi} \neq 0$ and $\overline{\psi} \cdot \overline{\psi}_0 = 0$).

Also, if $|\overline{\psi}| < \pi$, theorem 6 and 4 can be used to uniquely recover the screw dual vector $\overline{\psi}$ which is equivalent with computing $\log R$.

Theorem 9. The natural invariants $\alpha = \alpha + \varepsilon d, \overline{u} = \overline{u} + \varepsilon \overline{u}_0$ can be used to directly recover the structural invariants $Q$ and $\rho$ from (2.58):

$$Q = I + \sin \alpha \overline{u} + (1 - \cos \alpha) \overline{u}^2,$$
$$\rho = d\overline{u} + \sin \alpha \overline{u}_0 + (1 - \cos \alpha) \overline{u} \times \overline{u}_0.$$  

(2.70)

Proof. To prove (2.70) one can use equations (2.58) and (2.59). If these equations are equal then the structure of their dual parts lead to the result presented in (2.70).
2.2.2 Isomorphism between SE$_3$ and SO$_3$

This section highlights an isomorphism between the Lie group SE$_3$ and the Lie group SO$_3$. This isomorphism is the core of the simultaneous closed-form solution to the sensor calibration problem. First there are presented a set of theorems and remarks that will be used to map the sensor calibration problem from SE$_3$ into SO$_3$.

**Theorem 10.** The special Euclidean group (SE$_3$, ·) and (SO$_3$, ·) are connected via the isomorphism

$$\Phi : SE_3 \rightarrow SO_3, \quad \Phi(g) = (I + \varepsilon\tilde{\rho})Q,$$

where $g = \begin{bmatrix} Q & \rho \\ 0 & 1 \end{bmatrix}$, $\tilde{\rho}$ is the real antisymmetric tensor linked with the real vector $\rho$.

**Proof.** For any $g_1, g_2 \in SE_3$, the map defined in (2.71) yields:

$$\Phi(g_1 \cdot g_2) = \Phi(g_1) \cdot \Phi(g_2).$$

Let $R \in SO_3$. Based on theorem 5, which ensures a unique decomposition, the conclusion is that the only choice for $g$, such that $\Phi(g) = R$, is $g = \begin{bmatrix} Q & \rho \\ 0 & 1 \end{bmatrix}$. This underlines that $\Phi$ is a bijection and keeps all the internal operations.

**Remark 6.** The inverse of $\Phi$ is

$$\Phi^{-1} : SO_3 \leftrightarrow SE_3; \quad \Phi^{-1}(R) = \begin{bmatrix} Q & \rho \\ 0 & 1 \end{bmatrix},$$

where, $Q = Re(R)$, $\rho = vect(Du(R)Q^T)$.

Given two dual vectors $a$ and $b \in V_3$, $a \otimes b$ denotes a dual tensor called tensor (dyadic) product and is defined by:

$$a \otimes b : V_3 \rightarrow V_3, \quad (a \otimes b)v = (v \cdot b)a, \quad \forall v \in V_3.$$

(2.74)

Among the properties of (2.74) there are:

$$\text{trace } a \otimes b = a \cdot b,$$

$$\text{vect } a \otimes b = \frac{1}{2}b \times a,$$

$$\Phi(a \otimes b)(c \otimes d) = (b \cdot c)a \otimes d.$$
If \( B = \{ e_1, e_2, e_3 \} \) is a right-oriented orthonormal basis of dual vectors \((e_j \cdot e_j = \delta_{i,j} \text{ (Kronecker coefficients)}; e_1 \times e_2 = e_3)\) and \( a = \sum_{i=1}^{3} a^i e_i, b = \sum_{i=1}^{3} b^i e_i, a^i, b^i \in \mathbb{R}, i = 1,3, \)
the dyadic product \( a \otimes b \) is linked to a matrix of dual numbers computed as \([a \otimes b] = [a] [b]^T\),
where \([a] = [a^1 \ a^2 \ a^3]^T\) and \([b] = [b^1 \ b^2 \ b^3]^T\).

Also, the skew-symmetric tensor is linked to a matrix of dual numbers
\[
[a^\wedge] = \begin{bmatrix}
0 & -a^3 & a^2 \\
a^3 & 0 & -a^1 \\
a^2 & a^1 & 0
\end{bmatrix}.
\]

More details on relations between dual numbers, dual vectors and dual matrices can be found in [67, 5].

**Theorem 11** (Main theorem). Let \( \{b_1, b_2, b_3\} \) be a set of three dual vectors, which fulfills \( \text{Re} <b_1, b_2, b_3> \neq 0. \)
For any three dual vectors \( \{a_1, a_2, a_3\}, a_i \in V_3, i = 1,3, \)

(i) a unique dual tensor exists \( T \in L(V_3, V_3) \) such that \( T b_i = a_i, i = 1,3. \)

The computational formula for \( T \) is:
\[
T = \frac{a_1 \otimes (b_2 \times b_3)}{<b_1, b_2, b_3>} + \frac{a_2 \otimes (b_3 \times b_1)}{<b_1, b_2, b_3>} + \frac{a_3 \otimes (b_1 \times b_2)}{<b_1, b_2, b_3>};
\]

(ii)
\[
T \in SO_3 \Leftrightarrow \left\{ \begin{array}{c}
a_i \cdot a_j = b_i \cdot b_j; \quad i, j = 1,3 \\
< a_1, a_2, a_3 > = < b_1, b_2, b_3 >; \\
\text{Re} < b_1, b_2, b_3 > \neq 0
\end{array} \right\}
\]

Proof. Consider
\[
b^*_1 = \frac{b_2 \times b_3}{<b_1, b_2, b_3>}, \quad b^*_2 = \frac{b_3 \times b_1}{<b_1, b_2, b_3>}, \quad b^*_3 = \frac{b_1 \times b_2}{<b_1, b_2, b_3>},
\]
to be the reciprocal dual vectors of \( b_1, b_2, b_3 \) when \( \text{Re} <b_1, b_2, b_3> \neq 0. \)

Any dual vector \( v \in V_3 \) can be decomposed into \( v = \gamma_1 b_1 + \gamma_2 b_2 + \gamma_3 b_3, \) where \( \gamma_i = v \cdot b^*_i, i = 1,3. \) If the transformation induced by the dual tensor \( T \) ia applied over any \( v \in V_3, \)
the following result is obtained:

\[ Tv = \gamma_1 T b_1 + \gamma_2 T b_2 + \gamma_3 T b_3 \]
\[ = \gamma_1 a_1 + \gamma_2 a_2 + \gamma_3 a_3 \]
\[ = (v \cdot b_1^*) a_1 + (v \cdot b_2^*) a_2 + (v \cdot b_3^*) a_3 \]
\[ = (a_1 \otimes b_1^*) v + (a_2 \otimes b_2^*) v + (a_3 \otimes b_3^*) v \]

The previous equations give the internal structure of \( T \) which is:

\[ T = a_1 \otimes b_1^* + a_2 \otimes b_2^* + a_3 \otimes b_3^* \]

(2.82)

In order to prove \( (ii) \), first consider \( T \in SO_3 \). Based on \( (i) \), the following sequence of equalities is true

\[ a_i \cdot a_j = (T b_i) \cdot (T b_j) = b_j \cdot (T^T T b_i) = b_i \cdot b_j \]
\[ \langle a_1, a_2, a_3 \rangle = \langle T b_1, T b_2, T b_3 \rangle > \]
\[ = \det T \langle b_1, b_2, b_3 \rangle > \]
\[ = \langle b_1, b_2, b_3 \rangle > \]

(2.83)

thus proving the left to right implication.

Consider now that \[ \{a_i \cdot a_j = b_i \cdot b_j \} \]. From \[ a_j \cdot a_j = b_j \cdot b_j \] results that \[ a_i^* \cdot a_j = b_i^* \cdot b_j \] , where \[ a_i^*, b_j^* \], \( i, j = 1, 3 \) are defined by (2.80). If \( T = \sum_{i=1}^{3} a_i \otimes b_i^* \) then, as it is proved in [5], \[ T^T = \sum_{i=1}^{3} b_i^* \otimes a_i \]. This leads to \[ T T^T = I \], which combined with \[ \det T = \frac{\langle a_1, a_2, a_3 \rangle >}{\langle b_1, b_2, b_3 \rangle >} = 1 \] proves that \( T \in SO_3 \).

\( \square \)

Remark 7. If conditions (2.79) are true then \[ \{a_1, a_2, a_3\} \] and \[ \{b_1, b_2, b_3\} \] are bases in the free module of dual vectors and \[ \{a_1, a_2, a_3\} \] remains rigid in relation with \[ \{b_1, b_2, b_3\} \].

The proof of this Remark is detailed in [5]. In figure 2.2 is depicted a geometrical interpretation of the previous remark. Consider two sets of directed lines, each being parameterized by three dual vectors. The dual vectors \( b_j \times b_j \) are the common perpendicular for \( \{b_1, b_2\} \). If \( T b_i = a_i, i = 1, 3, T \in SO_3 \), the rigidity constraint implies that \( |b_i| = |a_i|, i = 1, 3 \). Also the common perpendiculars and dual angles between \( \{b_i, b_j\} \) are transferred to \( \{a_i, a_j\}, i, j = 1, 3, i \neq j \).
Next, the premises are built to have a generalized version of the main theorem. First, the properties of a dual tensor built using the dyadic product of $n$ dual vectors are discussed. Using this dual tensor the general form of theorem 11 can be developed.

**Remark 8.** Consider a set of dual vectors $\{b_1, b_2, ..., b_n\} \in V_3$. If $\exists i, j, k \in \{1, 2, ..., n\}$ with $Re(\langle b_i, b_j, b_k \rangle) \neq 0$ then the dual tensor

$$ S = b_1 \otimes b_1 + b_2 \otimes b_2 + ... + b_n \otimes b_n, \quad (2.84) $$

is symmetric and invertible.

**Proof.** The dual tensor $S = S + \varepsilon S_0$ is invertible if and only if $S = Re(S)$ is invertible [21, 67]. Taking into account the construction of $S$ results that its real part is $S = b_1 \otimes b_1 + b_2 \otimes b_2 + ... + b_n \otimes b_n$. Using the hypothesis $Re(\langle b_i, b_j, b_k \rangle) \neq 0$ results that $\langle b_i, b_j, b_k \rangle \neq 0$ which underlines that $S$ is invertible and thus the proof of the remark is completed.

$$\square$$

**Remark 9.** Consider a set of dual vectors $\{b_1, b_2, ..., b_n\} \in V_3$. If only two dual vectors that fulfill $Re(b_i \times b_j) \neq 0$, $i, j \in \{1, 2, ..., n\}$ are included in the set, the dual tensor $S$

$$ S = b_1 \otimes b_1 + b_2 \otimes b_2 + ... + b_n \otimes b_n + (b_i \times b_j) \otimes (b_i \times b_j), \quad (2.85) $$

is symmetric and invertible.
Proof. The triple scalar product of the dual vectors \( \{b_i, b_j, b_i \times b_j\} \) implies \( \text{Re}(<b_i, b_j, b_i \times b_j>) = \text{Re}(|b_i \times b_j|^2) \neq 0 \). This allows the application of Remark 7 over the dual vectors set \( \{b_1, b_2, \ldots, b_n, b_i \times b_j\} \), thus finalizing the proof.

Based on the previous remarks the **reciprocal dual vectors** of \( \{b_1, b_2, \ldots, b_n\} \), \( b_i \in V_3 \), \( i = \overline{1,n} \) are:

\[
b_i^* = S^{-1}b_i, i = \overline{1,n}.
\]

(2.86)

Using this definition it can be easily proven that

\[
b_1 \otimes b_1^* + b_2 \otimes b_2^* + \ldots + b_n \otimes b_n^* = I
\]

(2.87)

due to the transpose of:

\[
\sum_{i=1}^{n} b_i^* \otimes b_j = \sum_{i=1}^{n} [S^{-1}b_j] \otimes b_j = S^{-1}\sum_{i=1}^{n} b_i \otimes b_j = S^{-1}S = I.
\]

(2.88)

This observation allows that any dual vector \( v \in V_3 \) to be decomposed into

\[
v = \alpha_1 b_1 + \alpha_2 b_2 + \ldots + \alpha_n b_n
\]

(2.89)

where \( \alpha_j = v \cdot b_j^* \), \( i = \overline{1,n} \).

Now all the information needed to generalize the result presented in theorem 11 are available:

**Theorem 12.** Consider a set of dual vectors \( \{b_1, b_2, \ldots, b_n\} \), \( b_i \in V_3 \), \( i = \overline{1,n} \), \( n > 3 \) that includes three dual vectors that fulfill \( \text{Re} <b_i, b_j, b_k> \neq 0 \). For any set of dual vectors \( \{a_1, a_2, \ldots, a_n\} \), \( a_i \in V_3 \), \( i = \overline{1,n} \), the following two results are true:

(i) if a unique dual tensor \( T \in L(V_3, V_3) \) exists such that \( Tb_i = a_i \), \( i = \overline{1,n} \), the computational formula for \( T \) is:

\[
T = a_1 \otimes b_1^* + a_2 \otimes b_2^* + \ldots + a_n \otimes b_n^*.
\]

(2.90)

(ii)

\[
T \in SO_3 \Leftrightarrow \left\{ \begin{array}{l}
  a_j \cdot a_j = b_j \cdot b_j, i, j = \overline{1,n} \\
  <a_j, a_j, a_k> = <b_j, b_j, b_k>.
\end{array} \right.
\]

(2.91)

**Proof.** Consider a dual vector \( v \in V_3 \). If we apply the dual tensor \( T \) to \( v \) results that:

\[
Tv = T \sum_{i=1}^{n} \alpha_i b_i = \sum_{i=1}^{n} \alpha_i Tb_i = \sum_{i=1}^{n} <v, b_i^*> a_i = (\sum_{i=1}^{n} a_i \otimes b_i^*) v, \ \forall v \in V_3.
\]

(2.92)
This proves that $T = \sum_{i=1}^{n} a_i \otimes b_i^*$, which represents property (i) from the theorem.

In order to prove (ii), first consider $T \in SO_3$. The following sequence of equalities is true
\[ \forall i, j, k \in \overline{1, n}, \]
\[ a_j \cdot a_j = (T b_j) \cdot (T b_j) = b_j \cdot (T^T b_j) = b_j \cdot b_j, \quad i, j = 1, n \]
\[ <a_i, a_j, a_k> = <T b_i, T b_j, T b_k> \]
\[ = \det T <b_i, b_j, b_k> \]
\[ = <b_i, b_j, b_k> \]
(2.93)
thus proving the left to right implication.

Consider now that
\[ \begin{cases} a_j \cdot a_j = b_j \cdot b_j \\ <a_i, a_j, a_k> = <b_i, b_j, b_k> \end{cases} \]
If $T = \sum_{i=1}^{n} a_i \otimes b_i^*$ then $T^T = \sum_{k=1}^{n} b_k^* \otimes a_k$. This leads to $T^T T = \sum_{k=1}^{n} b_k^* \otimes a_k \sum_{i=1}^{n} a_i \otimes b_i^* = \sum_{k=1}^{n} \sum_{i=1}^{n} (a_k \cdot a_i)(b_i^* \otimes b_k^*)$. Using (44) results that $T^T T = \sum_{k=1}^{n} \sum_{i=1}^{n} (b_k \cdot b_i)(b_i^* \otimes b_k^*) = \sum_{k=1}^{n} b_k^* \otimes b_k \sum_{i=1}^{n} b_i \otimes b_i^* = I$. This result combined with
\[ \det T = <a_i, a_j, a_k> = 1 \] proves that $T \in SO_3$.

**Theorem 13.** Consider a set of dual vectors $\{b_1, b_2, ..., b_n\}, b_i \in V_3, \quad i = 1, n, \quad n > 3$ that includes at least two dual vectors that fulfill $\text{Re}(b_i \times b_j) \neq 0$. For any set of dual vectors $\{a_1, a_2, ..., a_n\}, a_i \in V_3, \quad i = 1, n$, the following two results are true:

(i) if a unique dual tensor $T \in L(V_3, V_3)$ exists in order to have: $T b_i = a_i, \quad i = 1, n + 1$, the dual tensor can be computed using:
\[ T = a_1 \otimes b_1^* + a_2 \otimes b_2^* + ... + a_n \otimes b_n^* + (a_i \times a_j) \otimes (b_i \times b_j)^* \quad (2.94) \]

(ii)
\[ T \in SO_3 \iff a_j \cdot a_j = b_j \cdot b_j, \quad i, j = 1, n + 1. \quad (2.95) \]

The notation $a_{n+1}, b_{n+1}$ the followings: $a_{n+1} = a_j \times a_j$ and $b_{n+1} = b_j \times b_j$.

**Proof.** The proof of this theorem in very similar to the proof of theorem 9. The only difference appears when proving that $\det T = 1$. The triple scalar product $<a_i, a_j, a_i \times a_j>$ which can be computed using $<a_i, a_j, a_i \times a_j> = \|a_i \times a_j\|^2 = \det T \|b_i \times b_j\|^2$. Taking into account that $\|b_i \times b_j\|^2 = \|b_i\|^2 \|b_j\|^2 - (b_i \cdot b_j)^2 = \|a_i\|^2 \|a_j\|^2 - (a_i \cdot a_j)^2 = \|a_i \times a_j\|^2$ it can be concluded that $\det T = 1$ which finalizes the proof. \(\square\)
Motion parameterization using dual Lie algebras

Fig. 2.3 Geometric representation of two sets of directed lines parameterized by dual vectors \( \{a_1, a_2, ..., a_n\}, \{b_1, b_2, ..., b_n\} \) which validate the rigidity constraints \( a_i \cdot a_j = b_i \cdot b_j, i, j = 1, n \).

Regarding geometrical interpretation, consider two sets of directed lines, represented as in figure 4, each directed line being parameterized by a dual vector. If \( T b_i = a_i, i = 1, n, T \in SO_3 \), the rigidity constraints implies that the set \( \{a_i\} \) is a rigid entity, the length of the common perpendiculars and dual angles between \( \{b_i, b_j\} \) being transferred to \( \{a_i, a_j\}, i, j = 1, n, i \neq j \) and \( |a_i| = |b_i| \), \( i = 1, n \).

2.3 Applications

2.3.1 Recovering motion parameters from feature-based description of rigid body

Working with rigid dual bases generates an increase in the flexibility of the rigid body motion parameterization because no assumptions of orthonormal reference frames is needed. Also, this approach is free of coordinates and only uses calculus with dual numbers and dual vectors, which implies a reduced number of algebraic equations and a more compact notation with fewer variables. Using point and line features of a rigid body, the dual rigid bases are constructed and new methods for computing orthogonal dual tensors, screw parameters and instantaneous screw parameters are proposed.

Let a rigid body be characterized by lines and points when the time function varies from an initial value \( t_0 \) to a floating value \( t \) (as shown in Figure 4). Regarding features combination, five cases can be considered to create a dual rigid basis:

1. Three non-coplanar lines
Consider that three non-coplanar lines \( l_i(t_0), i = 1, 3 \) are part of a rigid body. These lines can be characterized by normalized Plucker coordinates, which are staked in dual vectors \( u_i(t_0) = m_i(t_0) + \epsilon n_i(t_0), i = 1, 3 \). For each line the time \( t \) correspondent is denoted \( l_i(t), i = 1, 3 \) while the attached dual vectors are \( u_i(t) = m_i(t) + \epsilon n_i(t), i = 1, 3 \). In this case, the dual rigid bases are \( \{ e_{01}, e_{02}, e_{03} \} = \{ u_1(t_0), u_2(t_0), u_3(t_0) \} \) and \( \{ e_1, e_2, e_3 \} = \{ u_1(t), u_2(t), u_3(t) \} \).

2. Two non-parallel lines

If two lines are available, then the non-parallel condition must be fulfilled. Let \( l_1(t_0) \) and \( l_2(t_0) \) be the two lines that describe the rigid body at time \( t = t_0 \). The rigid basis \( \{ e_{01}, e_{02}, e_{03} \} \) is computed as \( \{ u_1(t_0), u_2(t_0), u_1(t_0) \times u_2(t_0) \} \). The same method is applied at any time \( t \) when, for example, \( l_1(t) \) and \( l_2(t) \) are valid.

3. Four non-planar points

Let \( P_i(t_0), i = 1, 4 \) be four non-planar points attached to a rigid body. The dual rigid basis \( \{ e_{01}, e_{02}, e_{03} \} \) is computed from

\[
\begin{align*}
\mathbf{e}_{01} &= \rho_{02} - \rho_{01} + \epsilon \rho_{01} \times \rho_{02} \\
\mathbf{e}_{02} &= \rho_{03} - \rho_{01} + \epsilon \rho_{01} \times \rho_{03} \\
\mathbf{e}_{03} &= \rho_{04} - \rho_{01} + \epsilon \rho_{01} \times \rho_{04}
\end{align*}
\]  

(2.96)

where \( \rho_{0i}, i = 1, 4 \) is the position vector of \( P_i(t_0) \). Using a similar procedure for \( P_i(t), i = 1, 4 \) the dual rigid basis \( \{ e_1, e_2, e_3 \} \) is obtained.
4. Three non-collinear points

In case of non-availability of one point from the four points set, if the remaining three points are non-collinear (e.g. \(P_1(t_0), P_2(t_0), P_3(t_0)\) are known) then \(e_{01}\) and \(e_{02}\) are computed using (2.96). For \(e_{03}\) the following equation is used:

\[
e_{03} = e_{01} \times e_{02}.
\] (2.97)

The same approach is valid for \(\{e_1, e_2, e_3\}\) generated by \(\{P_1(t), P_2(t), P_3(t)\}\).

5. One point and one line

Let the \(P_1(t_0)\) and \(l_1(t_0)\) be the available measurements on the rigid body. First, the constraint \(P_1(t_0) \notin l_1(t_0)\) must be fulfilled. If \(u = u + e u_0\) is the unit dual vector of \(l_1(t_0)\) then \(e_{01} = u, e_{02} = u \times w_0 + e \rho P_1(t_0) \times (u \times w_0)\) and \(e_{03} = e_{01} \times e_{02}\). The vector \(w_0\) is defined as \(w_0 = u_0 - \rho P_1(t_0) \times u\). The same approach is used to compute \(\{e_1, e_2, e_3\}\) when \(P_1(t)\) and \(l_1(t)\) are known.

The following result shows how dual rigid bases can be used to uniquely construct orthogonal dual tensors:

**Theorem 14.** If \(B = \{e_1, e_2, e_3\}\) is a dual rigid basis in \(V_3\) and \(e_i\) are continuous functions, then the dual tensor

\[
R = e_i \otimes e_i^0
\] (2.98)

is proper orthogonal and uniquely defined by:

\[
R(e_{0i}) = e_i, \ i = 1, 3.
\] (2.99)

**Proof.** From (2.99) and (2.98) it results that:

\[
R(e_{0i}) = (e_j \otimes e_j^0) e_{0i} = (e_{0i} \cdot e_j) e_j = \delta_i^j e_j = e_i
\] (2.100)

Let \(v \in V_3\) be an arbitrary vector function. Having \(v\) expressed in the basis \(B_0 = \{e_{01}, e_{02}, e_{03}\}\) as

\[
v = (v \cdot e_{0i}) e_{0i},
\] (2.101)

allows for

\[
Rv = (v \cdot e_{0i}) R e_{0i} = (v \cdot e_j^0) e_j = (e_j \otimes e_j^0) v,
\] (2.102)

which proves hypothesis (2.99).
Using (2.54) it results that
\[ R^T = e^i_0 \otimes e^i \]  \hspace{1cm} (2.103)
and, further on:
\[ RR^T = (e^i_0 \otimes e^i_0) (e^i_0 \otimes e^i_0) = (e^i_0 \cdot e^i_0) e^i_0 \otimes e^i_0. \] \hspace{1cm} (2.104)
From (2.32), combined with (2.31), the following relation emerges
\[ e^i \cdot e^j = e^i_0 \cdot e^j_0, \quad i, j = 1, 3, \] \hspace{1cm} (2.105)
and thus (2.104) becomes:
\[ RR^T = (e^i \cdot e^j) e^i_0 \otimes e^j_0 = e^i_0 \otimes e^i_0 = I. \] \hspace{1cm} (2.106)
This leads to \( det RR^T = 1 \) and \( det R^2 = 1 \), thus resulting \( det R \in \{-1, 1\} \). From (2.98) and (2.47) the \( det R \) can be computed as:
\[ det R = \frac{<e^1, e^2, e^3>}{<e^0_1, e^0_2, e^0_3>}. \] \hspace{1cm} (2.107)
Because \( e^k_0, k = 1, 3 \) are continuous functions, it results that
\[ \lim_{t \to t_0} e^k(t) = e^0_k, \] \hspace{1cm} (2.108)
which leads to
\[ det R = 1, \] \hspace{1cm} (2.109)
and proves that \( R \) is a proper orthogonal dual tensor. \( \square \)

**Remark 10.** For any orthogonal dual tensor defined as in (2.98), the values of \( Q \in SO_3 \) and \( \hat{\rho} \in so_3 \) can be recovered using the procedure detailed next. If \( e_i = e^i + \varepsilon e^i_0 \) and \( e^i_0 = e^i_0 + \varepsilon e^i_0 \), then
\[ R = e^i \otimes e^i_0 + \varepsilon[e^i \otimes e^i_0 + e^i_0 \otimes e^i] \] \hspace{1cm} (2.110)
which leads to \( Q = e^i \otimes e^i_0 \) and \( \hat{\rho} = Q_0 (e^i_0 \otimes e^i_0) \), where \( Q_0 = e^i \otimes e^i_0 + e^i_0 \otimes e^i \).

Based on the dual tensors properties, \( \hat{\rho} \) can also be computed using \( \hat{\rho} = (Q_0 e^i_0) \otimes e^i_0 \), which gives the recovering formula of \( \rho = vect \hat{\rho} \):
\[ \rho = \frac{1}{2} e^i \times Q_0 e^i_0. \] \hspace{1cm} (2.111)
Screw Parameters generated by Dual Rigid Bases

Next, a new method for computing the screw parameters is presented. The solution is based on an orthogonal dual tensor, that models the rigid body motion, generated by dual rigid bases.

Remark 11. Using (2.55), (2.56), and (2.98) the following equations emerge:

\[ u \sin \alpha = \frac{1}{2} \mathbf{e}_0^j \times \mathbf{e}_i, \quad (2.112) \]

\[ \cos \alpha = \frac{1}{2} \mathbf{e}_0^j \cdot \mathbf{e}_i - 1. \quad (2.113) \]

Equations (2.112) and (2.113) solve the problem of finding the logarithm of the orthogonal dual tensor, while the following theorem gives a computational procedure of finding \( u \) and \( \alpha \) without the computation of the dual tensor.

Theorem 15. The natural invariants of the proper orthogonal tensor \( \mathbf{R} = \mathbf{e}_i \otimes \mathbf{e}_0^j \) are

\[
\mathbf{u} = \begin{cases} 
\frac{1}{2} \mathbf{e}_0^j \times \mathbf{e}_i, & \text{Re}(\mathbf{e}_0^j \times \mathbf{e}_i) \neq 0 \\
\frac{1}{2} \mathbf{e}_0^j \cdot \mathbf{e}_i - 1, & \text{Re}(\mathbf{e}_0^j \cdot \mathbf{e}_i) = 0 \text{ and } \text{Re}(\mathbf{e}_0^j \times \mathbf{e}_i) = -1 \\
\frac{1}{2} \mathbf{e}_0^j \cdot \mathbf{e}_i + \frac{1}{2} \mathbf{e}_0^j \times \mathbf{e}_i, & \text{Re}(\mathbf{e}_0^j \times \mathbf{e}_i) = 0 \text{ and } \text{Re}(\mathbf{e}_0^j \cdot \mathbf{e}_i) = 3 
\end{cases}
\]

\[
\alpha = \arctan2\left( \frac{\frac{1}{2} \mathbf{e}_0^j \times \mathbf{e}_i - 1}{\frac{1}{2} \mathbf{e}_0^j \cdot \mathbf{e}_i - 1} \right), \quad (2.114)
\]

where \( \rho \) is given by (2.111) and the sum \( \mathbf{e}_0^j + \mathbf{e} \) is properly chosen to maximize norm of any of the three sums \( \mathbf{e}_0^j + \mathbf{e}_i, i = 1, 3 \).

Proof. Consider \( \mathbf{B}_0 = \{ \mathbf{e}_{01}, \mathbf{e}_{02}, \mathbf{e}_{03} \} \) a dual rigid basis and \( \mathbf{B}_0^* = \{ \mathbf{e}_{10}, \mathbf{e}_{20}, \mathbf{e}_{30} \} \) its reciprocal dual basis. A correspondent of \( \mathbf{B}_0 \) at a generic time \( t \) will be denoted by \( \mathbf{B} = \{ \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \} \).

Three types of motion cases can be considered:

- if \( \text{Re}(\mathbf{e}_0^j \times \mathbf{e}_i) \neq 0 \) then a general Mozzi-Chasles type motion is active;
- if \( \text{Re}(\mathbf{e}_0^j \times \mathbf{e}_i) = 0 \text{ and } \text{Re}(\mathbf{e}_0^j \cdot \mathbf{e}_i) = -1 \) then the Mozzi-Chasles motion is in fact a translation combined with a rotation of an angle equal to \( \pi \);
- if \( \text{Re}(\mathbf{e}_0^j \times \mathbf{e}_i) = 0 \text{ and } \text{Re}(\mathbf{e}_0^j \cdot \mathbf{e}_i) = 3 \) then the Mozzi-Chasles motion represents a pure translation;
2.3 Applications

For \( \text{Re}(e_0^i \times e_j) \neq 0 \) the combination between (2.112) and Theorem 2 gives the values of \( u \) and \( \alpha \). When \( \text{Re}(e_0^i \times e_j) = 0 \) and \( \text{Re}(e_0^i \cdot e_j) = -1 \) the solution is given by any of the three vectors \( e_{0i} + e_i \). In order to avoid singularities we will choose \( e_0 + e \) to be the pair \( e_{0i} + e_i, i = 1, 3 \) with the maximum norm. The validation of (2.114) and (2.115) is provided by (2.112), (2.113) and Remark 10.

The result illustrated by Theorem 15 represents a new algebraic technique for computing the screw parameters of a rigid body motion. This technique is based on dual rigid bases and illustrates all the cases that can occur during screw parameters detection.

Instantaneous Screw Parameters generated by Dual Rigid Bases

In this subsection a new dual rigid bases based method for computing the instantaneous screw parameters is presented.

If \( R \) is defined by (2.98) then
\[
\dot{R} = \dot{e}_i \otimes e_0^i, \tag{2.116}
\]
which together with (2.116) and (2.103) give:
\[
\Phi = (e_0^i \otimes e_0^j)(e_0^i \otimes e_i) = (e_0^i \cdot e_0^j)(e_i \otimes e_j). \tag{2.117}
\]

Based on Remark 3 results that
\[
\Phi = (e^i \cdot e^j)(\dot{e}_i \otimes e_j) = \dot{e}_i \otimes [(e^i \cdot e^j)e_j] = \dot{e}_i \otimes e^i, \tag{2.118}
\]
Denoting by \( \Phi = \overline{\omega} \), leads to the following representation for the angular velocity dual tensor:
\[
\overline{\omega} = \dot{e}_i \otimes e^i. \tag{2.119}
\]
If \( \omega = \text{vect}\overline{\omega} \) then the following theorem can be stated:

**Theorem 16.** If a dual rigid basis \( B = \{e_1, e_2, e_3\} \) is given, then a dual vector \( \omega \) exists and is expressed by
\[
\omega = \frac{1}{2} e^i \times \dot{e}_i \tag{2.120}
\]
so that
\[
\dot{e}_i = \omega \times e_i. \tag{2.121}
\]

**Proof.** Combining \( \omega \times e_j = \overline{\omega} e_j \) with \( \overline{\omega} = \dot{e}_j \otimes e_j \) results that \( \omega \times e_j = (e_0^j \otimes e_0^j)e_j \). Using (2.74), the definition of the dyadic product, we can conclude that \( \omega \times e_j = (e_0^j \cdot e_0^j)e_j = \delta^j_0 e_j = \dot{e}_j \).
Using basic dual vectors calculus, equation (2.120) can be transformed into
\[
\omega = \left( \mathbf{e}_2 \cdot \mathbf{e}_3 \right) \mathbf{e}_1 + \left( \mathbf{e}_3 \cdot \mathbf{e}_1 \right) \mathbf{e}_2 + \left( \mathbf{e}_1 \cdot \mathbf{e}_2 \right) \mathbf{e}_3,
\]
which has the advantage of computing the dual angular velocity vector without involving the reciprocal dual basis.

The dual angular velocity vector \( \omega \) completely characterize, at a certain time, the velocity field of an rigid body in motion. Let \( \mathbf{v}_P \) be the linear velocity of a point described by the position vector \( \mathbf{\rho}_P \):
\[
\mathbf{v}_P = \mathbf{v} + \omega \times \mathbf{\rho}_P.
\]

Based on Theorem 2, for \( ||\omega|| \neq 0 \) the instantaneous screw axis unit dual vector is
\[
u_{\omega} = \frac{\omega}{||\omega||} + \varepsilon \frac{\mathbf{v} \times \omega}{||\omega||^3}
\]
and
\[
||\omega|| = ||\omega|| + \varepsilon \frac{\mathbf{v} \cdot \omega}{||\omega||}.
\]

For \( \mathbf{v} \cdot \omega = \mathbf{v}_P \cdot \omega = v_{\text{min}} \), the magnitude of \( \omega \) is \( ||\omega|| = ||\omega|| + \varepsilon v_{\text{min}} \). If \( p = \frac{\mathbf{v} \cdot \omega}{||\omega||^2} = v_{\text{min}}/||\omega|| \) denote the pitch of the screw axis [64] then \( ||\omega|| = ||\omega||\left(1 + \varepsilon p\right) \). For \( ||\omega|| = 0 \) the structure of the dual angular velocity vector is \( \omega = \varepsilon \mathbf{v} \), which represents an instantaneous pure translation.

Equation (2.122) gives a new compact form for \( \omega \), which can also be written as
\[
\omega = K_1 \mathbf{e}_1 + K_2 \mathbf{e}_2 + K_3 \mathbf{e}_3,
\]
where the dual tensors \( K_i, i = 1, 3 \) are \( K_1 = \frac{\mathbf{e}_3 \otimes \mathbf{e}_2}{\langle \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \rangle}, K_2 = \frac{\mathbf{e}_1 \otimes \mathbf{e}_3}{\langle \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \rangle}, K_3 = \frac{\mathbf{e}_2 \otimes \mathbf{e}_1}{\langle \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \rangle} \).

Next we present the algorithms (pseudo code version) that put the methods, from section 4, into practice. Also, symbolic and numerical computational examples are presented and discussed.

Once the time variation laws of the features are established, the orthogonal dual tensor is computed using Algorithm 1.

The input data are the coordinates of three points denoted \( \mathbf{\rho}_{01}, \mathbf{\rho}_{02}, \mathbf{\rho}_{03} \) and the coordinates of their corespondents \( \mathbf{\rho}_1, \mathbf{\rho}_2, \mathbf{\rho}_3 \) at a certain moment in time. The dual rigid basis \( \{ \mathbf{e}_{01}, \mathbf{e}_{02}, \mathbf{e}_{03} \} \) is computed from (2.96) and (2.97). Using a similar procedure for \( \mathbf{\rho}_i, i = 1, 3 \), the dual rigid basis \( \{ \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \} \) is obtained. At this point a test is necessary to establish if the compatibility (Definition 1) condition is fulfilled. The reciprocal basis \( \{ \mathbf{e}_0^1, \mathbf{e}_0^2, \mathbf{e}_0^3 \} \) is computed using equation (2.31).
Algorithm 1: Orthogonal dual tensors generated by dual rigid bases

1. Compute the dual rigid bases \( \{e_1(t), e_2(t), e_3(t)\} \) and \( \{e_{01}, e_{02}, e_{03}\} \), as presented for any of the five features combinations. Verify the rigidity condition.
2. Compute the reciprocal dual basis \( \{e_0^1, e_0^2, e_0^3\} \)
3. Extract the orthogonal dual tensor using
   \[
   R = e_i(t) \otimes e_0^i = e_i(t)e_0^i, \quad i = 1, 3
   \]

Algorithm 2: Screw parameters generated by dual rigid bases

1. Compute the dual rigid bases \( \{e_{01}, e_{02}, e_{03}\} \) and \( \{e_1, e_2, e_3\} \)
2. Compute the reciprocal dual basis \( \{e_0^1, e_0^2, e_0^3\} \)
3. \( \alpha = \arctan \left( \pm \frac{1}{2} |e_0^i \times e_i|, \frac{1}{2} (e_0^i \cdot e_i - 1) \right) \).
4. \( \text{if } \text{Re} \left( \frac{1}{2} e_0^i \times e_i \right) \neq 0 \text{ then} \)
5. \( u = \pm \frac{e_0^i \times e_i}{|e_0^i \times e_i|} \) \quad (2.129)
6. \( \text{end} \)
7. \( \text{else} \)
8. \( \rho = \frac{1}{2} e_i \times Q_0 e_0^i, \text{ where } Q_0 = e_i \otimes e_0^i + e_i^* \otimes e_0^i, \quad e_i = e_i + \varepsilon e_i^*, \quad e_0^i = e_0^i + \varepsilon e_0^i; \)
9. \( \text{if } \text{Re} \left( \frac{1}{2} e_0^i \cdot e_i \right) = -1 \text{ then} \)
10. \( \text{Compute } e_0 + e \text{ as the pair } \{e_0, e_i\}, \quad i = 1, 3 \text{ with the maximum norm} \)
11. \( u = \frac{e_0 + e}{|e_0 + e|} + \varepsilon \frac{1}{2} \rho \times \frac{e_0 + e}{|e_0 + e|} \) \quad (2.130)
12. \( \text{end} \)
13. \( \text{if } \text{Re} \left( \frac{1}{2} e_0^i \cdot e_i \right) = 3 \text{ then} \)
14. \( u = \frac{\rho}{|\rho|} \) \quad (2.131)
15. \( \text{end} \)
16. \( \text{end} \)
Algorithm 3: Instantaneous screw parameters generated by dual rigid bases

1 Coordinates of three points $\rho_1, \rho_2, \rho_3$ and their velocities $v_1, v_2, v_3$ are chosen to be the input data. Before continuing a compatibility test must be carried:

$$(\rho_1 - \rho_i) \cdot (v_j - v_1) + (\rho_1 - \rho_j) \cdot (v_i - v_1) = 0; \ i, j = 2, 3, i \neq j.$$  

2 The dual rigid basis $\{e_1, e_2, e_3\}$ is computed using

$$e_1 = \rho_2 - \rho_1 + \varepsilon \rho_1 \times \rho_2; \ e_2 = \rho_3 - \rho_1 + \varepsilon \rho_1 \times \rho_3; \ e_3 = e_1 \times e_2.$$  

3 In order to generate the values of $\dot{e}_i$, the following equations are used:

$$\dot{e}_1 = v_2 - v_1 + \varepsilon (v_1 \times \rho_2 - v_2 \times \rho_1); \ \dot{e}_2 = v_3 - v_1 + \varepsilon (v_1 \times \rho_3 - v_3 \times \rho_1),$$  

$$\dot{e}_3 = \dot{e}_1 \times e_2 + e_1 \times \dot{e}_2.$$  

4 Once $e_i$ and $\dot{e}_i$ are obtained, the computation of ISA is a direct process based on:

$$\omega = \left(\frac{e_2 \cdot e_3}{e_1} \right) e_1 + \left(\frac{\dot{e}_3 \cdot e_1}{e_2} \right) e_2 + \left(\frac{\dot{e}_1 \cdot e_2}{e_3} \right) e_3.$$  

5 if $||\omega|| \neq 0$ then

$$u_\omega = \frac{\omega}{||\omega||} + \frac{\varepsilon \omega \times (v \times \omega)}{||\omega||^3}; \ ||\omega|| = ||\omega|| + \varepsilon \frac{v \cdot \omega}{||\omega||}.$$  

6 end

7 else

$$u_\omega = \frac{v}{||v||} + \varepsilon w \times \frac{v}{||v||}; \forall w \in V_3; \ ||\omega|| = \varepsilon ||v||.$$  

8 end

The algorithm for computing the natural invariants of a proper orthogonal dual tensor is composed from intuitive and easy to implement steps (Algorithm 2). The case of three points is discussed, but the algorithm can be easily adapted for any type of measurements which can be used to construct dual rigid bases.

The third algorithm aims to implement the method proposed for the computation of the instantaneous screw axis. Usually, the lowest number of points needed to solve the ISA recovery problem is three. Thus, in Algorithm 3 this case is detailed.
2.3 Applications

2.3.2 A different approach to solving the AX=XB sensor calibration problem

The sensor calibration problem has been, in the last decades, one of the most interesting challenge in the field of applied robotics. Different techniques were proposed to solve the classic $AX = XB$ problem. Depending on the outcome, the proposed techniques can be divided into: separable [68, 69], simultaneous [70] or iterative [71–73]. Any method used to solve the sensor calibration problem presents its own advantages and disadvantages [74]. The separable solutions are simple and fast solutions; however, errors accumulated from the attitude component could be transferred to the positional component. As a result, simultaneous solutions were developed. However, these solutions produce variable results depending on the scaling of the positional component. To weight the attitude and positional components, iterative methods were created. Even if these solutions are often more accurate, their complexity depends on the starting criteria. In addition, there is generally no guarantee that the convergent solution is the optimal one. Based on desired accuracy and complexity, users must decide which type of method to use for their applications. In this context this research is focused on methods that can both provide the necessary and sufficient conditions that ensure the existence and uniqueness of solutions and generate new simultaneous closed-form solutions to the $AX = XB$ sensor calibration problem. The practical side of these type of solutions is usually linked with the hand-eye calibration problem. Until now, the methods proposed to solve this type of problem vary from dual quaternions to global polynomial optimization [75–80].

Compatibility conditions were drawn by Chen in [70] using screw theory. These compatibility conditions can be used to establish if a certain set of recorded motions is proper to find the solution to the sensor calibration problem. Also, these compatibility conditions can be interpreted as rigidity constraints. In [75] Daniilidis used dual quaternions to give an algebraic interpretation to Chen’s approach. The Kronecker product is used by Andreff et al. in [81] to solve the sensor problem by reformulating it into a linear system.

In the last years there were several techniques proposed for solving the $AX = XB$ problem. Ackerman et al. [82] proposed a method based on Euclidean group invariants. After describing two algorithms used to select the corresponding pairs of homogeneous matrices $(A,B)$ from data with unknown correspondences, the authors give the necessary and sufficient conditions for the unique solution to the sensor calibration problem. These conditions are similar to the ones presented in [70]. In [83], the same authors analyze a probabilistic technique that can be used to solve the $AX = XB$ problem.
Motion parameterization using dual Lie algebras

Usually the sensor calibration problem

\[ AX = XB \]  \hspace{1cm} (2.135)

is solved in \( SE_3 \). Being elements of \( SE_3 \), \( A, X \) and \( B \) are each homogeneous transformations with \( A \) and \( B \) given from sensor measurements, and \( X \) unknown. In this section we present a new approach of finding the solutions to the sensor calibration problem by making use of an isomorphism between \( SE_3 \) and \( SO_3 \). Thus the new form of the sensor calibration problem is

\[ AX = XB \]  \hspace{1cm} (2.136)

where \( A = \Phi(A), B = \Phi(B), X = \Phi(X) \), \( \Phi \) being the isomorphism from theorem 10. The detailed proofs of the theorems presented next can be found in [1].

Let \( a \in \log A \) and \( b \in \log B \). Consider \( a = \text{vect} \bar{a} = \theta_a u_a \) the screw dual vector of \( A \) and \( b = \text{vect} \bar{b} = \theta_b u_b \) the screw dual vector of \( B \). Different cases can be analyzed, each being related to the number of rigid displacements recorded by the robot-sensor ensemble.

The following theorem gives the \( SO_3 \) inversion form to the \( AX = XB \) problem:

**Theorem 17.** If the equation

\[ a = Xb \]  \hspace{1cm} (2.137)

admits the solution \( X \in SO_3 \) then \( X \) is also the solution of (2.136), where \( A = e^{\bar{a}} \) and \( B = e^{\bar{b}} \).

**Proof.** Taking into account (2.66), the following series of algebraic identities are true:

\[ e^{2a} = e^{Xb} \iff e^{a} = e^{XbX^T} \iff e^{a} = X e^{bX^T} \]. The last identity leads to \( e^{a} X = X e^{bX^T} \) and proves that \( X \) is also the solution of \( AX = XB \). If equation (2.137) is solvable, by applying \( \Phi^{-1} \) on its solution we obtain the solution to equation (2.135). \( \square \)

**Theorem 18.** The solution \( X \in SO_3 \) to equation (2.136), when a single motion is recorded, can be computed if and only if \( |a| = |b| \iff |\theta_a| = |\theta_b| \). The analytical form of \( X \) is:

\[ X = R^# R_b = R_a R^# \]  \hspace{1cm} (2.138)

where \( R^# \) can be computed as presented in [1].

Considering \( \theta_a = \theta_a + \varepsilon d_a \) and \( \theta_b = \theta_b + \varepsilon d_b \), the condition \( |\theta_a| = |\theta_b| \) can also be expressed as \( |\theta_a| = |\theta_b| \) and \( (\text{sign} \theta_a) d_a = (\text{sign} \theta_b) d_b \). These are equivalent with the compatibility conditions presented in [70], which were later reformulated using dual quaternions in [75].
2.3 Applications

If more than one general motion is recorded, the sensor calibration problem will have a unique complete closed-form solution. Consider \( \mathbf{a}_i, \mathbf{b}_i \) to be the screw dual vectors of \( A_i, B_i \in SO_3 \ i = 1,n \).

**Theorem 19.** Consider three rigid motions to be recorded by a robot-sensor ensemble. The system of equations

\[
A_i X = X B_i, \quad A_i, B_i \in SO_3 \quad i = 1,3
\]

admits a unique solution \( X \in SO_3 \) if and only if

\[
\left\{ \begin{array}{l}
\mathbf{a}_i \cdot \mathbf{a}_j = \mathbf{b}_i \cdot \mathbf{b}_j, \quad i, j = 1,3 \\
< \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3 > = < \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3 > \\
Re < \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3 > \neq 0
\end{array} \right.
\]

The algebraic form of the solution is:

\[
X = \mathbf{a}_1 \otimes \mathbf{b}^*_1 + \mathbf{a}_2 \otimes \mathbf{b}^*_2 + \mathbf{a}_3 \otimes \mathbf{b}^*_3,
\]

where \( \mathbf{b}^*_i \) are the reciprocal dual vectors of \( \mathbf{b}_i \).

The compatibility constraints from the previous theorem are equivalent with the fact that \( \{ \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3 \}, \{ \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3 \} \) have the property of being rigid bases of dual vectors as presented in Remark 7.

**Theorem 20.** If only two rigid motions are recorded by the robot-sensor ensemble, the system

\[
A_i X = X B_i, \quad A_i, B_i \in SO_3 \quad i = 1,2
\]

has a unique solution

\[
X \in SO_3 \iff \left\{ \begin{array}{l}
\mathbf{a}_i \cdot \mathbf{a}_j = \mathbf{b}_i \cdot \mathbf{b}_j, \quad i, j = 1,2 \\
Re(\mathbf{b}_1 \times \mathbf{b}_2) \neq 0
\end{array} \right.
\]

The solution is:

\[
X = \mathbf{a}_1 \otimes \mathbf{b}^*_1 + \mathbf{a}_2 \otimes \mathbf{b}^*_2 + (\mathbf{a}_1 \times \mathbf{a}_2) \otimes (\mathbf{b}_1 \times \mathbf{b}_2)^*.
\]

The conditions (2.140) and (2.142) include the constraints given in [82], which are necessary for the existence of the solution. Moreover, equations (2.140) and (2.140) include both the necessary and sufficient conditions to ensure the existence of a unique solution.
Consider now a number of robot-sensor ensemble rigid motions higher than three. For the first time, an algebraic closed form solution can be drawn for the general case of the sensor calibration problem:

**Theorem 21.** For a set of rigid motions recorded by a robot-sensor ensemble, the system

\[ A_iX = XB_i, \quad A_i, B_i \in SO_3, \quad i = 1, n \]

has a unique solution \( X \in SO_3 \) if and only if

\[
\begin{cases}
\mathbf{a}_i \cdot \mathbf{a}_j = \mathbf{b}_i \cdot \mathbf{b}_j, & i, j = 1, n + 1 \\
\exists i, j \in 1, n \text{ in order to have } \text{Re}(\mathbf{b}_i \times \mathbf{b}_j) \neq 0
\end{cases}
\]

(2.144)

We have denoted by \( \mathbf{a}_{n+1}, \mathbf{b}_{n+1} \) the followings: \( \mathbf{a}_{n+1} = \mathbf{a}_i \times \mathbf{a}_j \) and \( \mathbf{b}_{n+1} = \mathbf{b}_i \times \mathbf{b}_j \).

If \( \exists i, j, k \in \{1, 2, ..., n\} \) with \( \langle \mathbf{b}_i, \mathbf{b}_j, \mathbf{b}_k \rangle = \langle \mathbf{a}_i, \mathbf{a}_j, \mathbf{a}_k \rangle \) and \( \text{Re}(\langle \mathbf{b}_i, \mathbf{b}_j, \mathbf{b}_k \rangle) \neq 0 \) the solution is

\[ X = \sum_{i=1}^{n} \mathbf{a}_i \otimes \mathbf{b}_i^* \]  

(2.145)

else the solution is

\[ X = \sum_{i=1}^{n+1} \mathbf{a}_i \otimes \mathbf{b}_i^* \]  

(2.146)

**Proof.** The proof of this theorem is a direct application of theorem 15. \( \square \)

The last results solves the problem of finding the closed-form simultaneous solution to the general sensor calibration problem. The compatibility conditions (94) that need to be fulfilled in order to have a solution can be geometrically interpreted as follows

- at least two screw axes of the robot and sensor need to be nonparallel
- to have the same dual angle between \( (\mathbf{a}_i, \mathbf{a}_j) \) and \( (\mathbf{b}_i, \mathbf{b}_j) \), with \( i, j = 1, n \) which implies having the same length of the common perpendiculars and angles between \( (\mathbf{a}_i, \mathbf{a}_j) \) and \( (\mathbf{b}_i, \mathbf{b}_j) \), with \( i, j = 1, n \)
- the rigid motions of both robot and sensor have the same screw dual angle due to \( |\mathbf{a}_i| = |\mathbf{b}_i|, \quad i = 1, n \) (same rotation and translation)

These conditions are necessary and sufficient, ensure a unique solution to the sensor calibration problem and generalize the compatibility conditions presented in [82, 73].

Regarding the implementation of the proposed solutions, a problem emerges when working with data acquired by different sensors. The previous equations cannot be directly
applied because if the rigidity constraints are not fulfilled. Thus, in order to deal with noisy input data a filtering procedure is needed. Next we present two filtering techniques and the algorithm that can be used to put into practice the solutions proposed in the previous section.

• **QR filtering procedure**

The following filtering procedure is an adjustment over the classical Gram-Schmidt algorithm. The result of the filtering procedure is similar to the QR decomposition used in [67, 84]. Consider \( \mathbf{c}_i, i = 1, 3 \), to be the columns of the dual tensor \( \mathbf{R} \) which was built from noisy measurements. The correction procedure performs the Gram-Schmidt algorithm but does not divide any dual vector by its length until the end of the algorithm. This approach allows the algorithm to avoid multiple problems that can emerge when multiplying a dual vector with the inverse of a dual number.

The procedure starts by setting:

\[
\mathbf{y}_1 = \mathbf{c}_1.
\]

(2.147)

Next \( \mathbf{y}_2 \) is computed from

\[
\mathbf{y}_2 = \mathbf{c}_2 - \left( \frac{\mathbf{c}_2 \cdot \mathbf{y}_1}{\mathbf{y}_1 \cdot \mathbf{y}_1} \right) \mathbf{y}_1,
\]

(2.148)

while \( \mathbf{y}_3 \) is generated by:

\[
\mathbf{y}_3 = \mathbf{c}_3 - \left( \frac{\mathbf{c}_3 \cdot \mathbf{y}_1}{\mathbf{y}_1 \cdot \mathbf{y}_1} \right) \mathbf{y}_1 - \left( \frac{\mathbf{c}_3 \cdot \mathbf{y}_2}{\mathbf{y}_2 \cdot \mathbf{y}_2} \right) \mathbf{y}_2.
\]

(2.149)

The resulting dual vectors \( \mathbf{y}_i, i = 1, 3 \) are orthogonal but their magnitude is not equal to \( 1 + \varepsilon 0 \) so the final step is:

\[
\mathbf{c}_i^\dagger = \frac{\mathbf{y}_i}{|\mathbf{y}_i|}, i = 1, 3.
\]

(2.150)

In the end we recover \( \mathbf{R}^\dagger = [\mathbf{c}_1 \mathbf{c}_2 \mathbf{c}_3] \in \mathbb{S} \mathbb{O}_3 \).

• **SVD filtering procedure**

Let \( \mathbf{R} = [\mathbf{c}_1 \mathbf{c}_2 \mathbf{c}_3] \) be the dual matrix attached to the dual tensor \( \mathbf{R} \). This dual matrix has a series of properties which can be used to filter noisy data. Its singular value decomposition (SVD) should be [67]

\[
\mathbf{R} = \mathbf{U} \Sigma \mathbf{V}^T,
\]

(2.151)

where \( \Sigma = \text{diag}\{\sigma_1, \sigma_2, \sigma_3\} \), and \( \mathbf{U}, \mathbf{V} \in \mathbb{O}_3 \), \( \mathbf{U} \mathbf{U}^T = \mathbf{I}, \mathbf{V} \mathbf{V}^T = \mathbf{I} \).

After SVD is performed, the orthogonal dual matrix that will be used to carry out future experiments is build using:

\[
\mathbf{R}^{d\dagger} = \mathbf{U} \text{diag}\{1, 1, \det \mathbf{U} \cdot \det \mathbf{V}\} \mathbf{V}^T, \quad \mathbf{R}^{d\dagger} \in \mathbb{S} \mathbb{O}_3
\]

(2.152)
**Algorithm 4:** Computational procedure for the solution proposed for the $AX = XB$ sensor calibration problem

1. Input data $A_i, B_i \in SE_3, i = 1 \ldots n$
2. Apply the isomorphism (2.71) to obtain the dual tensors $\mathbf{A}_i = \Phi(A_i), \mathbf{B}_i = \Phi(B_i)$
3. Compute $\mathbf{u}_i = vect(log(A_i)), \mathbf{b}_i = vect(log(B_i))$ using equations from theorem 3 or theorem 4.
4. The solution to $\mathbf{A}_i \mathbf{X} = \mathbf{X} \mathbf{B}_i \iff \mathbf{A}_i \mathbf{X} = \mathbf{X} \mathbf{B}_i$ depends on the number of recorded motions.
   - If $n = 1$ and $|\mathbf{a}_i| = |\mathbf{b}_i|$ then an infinite set of solutions is valid. A particular solution $\mathbf{X}^\#$ can be computed as presented in [1]. The particular solution $\mathbf{X}^\#$ is from $SO_3$ and the general solution is $\mathbf{X} = \Phi^{-1}(\mathbf{X}^\# R_i), \mathbf{X} \in SE_3$. For $|\mathbf{a}_i| \neq |\mathbf{b}_i|$ the algorithm cannot generate a solution.
   - For $n \geq 2$ we need to follow the next steps:
5. Compute the reciprocal dual vectors $\mathbf{b}_i^* = S^{-1} \mathbf{b}_i, i = 1 \ldots n$, using equation (2.86).
6. Compute the solution of the $\mathbf{A}_i \mathbf{X} = \mathbf{X} \mathbf{B}_i$ using equations (2.145) or (2.146).
7. Apply one of the two filtering procedures (QR or SVD) to recover $\mathbf{X}^\dagger$ or $\mathbf{X}^{\dagger \dagger}$
8. Output data $\mathbf{X}^\dagger = \Phi^{-1}(\mathbf{X}^{\dagger}), \mathbf{X}^\dagger \in SE_3$ or $\mathbf{X}^{\dagger \dagger} = \Phi^{-1}(\mathbf{X}^{\dagger \dagger}), \mathbf{X}^{\dagger \dagger} \in SE_3$.

In order to evaluate the bias of the solution given by the proposed algorithm (solution which from now on will be denoted by $\mathbf{X}_{alg} = \Phi^{-1}(\mathbf{X}_{alg})$) in comparison with the actual solution (which will from now on will be denoted by $\mathbf{X}_{act} = \Phi^{-1}(\mathbf{X}_{act})$) we first need to compute two dual vectors: $\mathbf{y}_{alg} = vect(log(\mathbf{X}_{alg})) = (\alpha_{alg} + \epsilon d_{alg}) \mathbf{u}_{alg}$ and $\mathbf{y}_{act} = vect(log(\mathbf{X}_{act})) = (\alpha_{act} + \epsilon d_{act}) \mathbf{u}_{act}$. The dual angle between $\mathbf{y}_{alg}$ and $\mathbf{y}_{act}$

$$
\epsilon_1 = atan2(|\mathbf{u}_{alg} \times \mathbf{u}_{act}|, \mathbf{u}_{alg} \cdot \mathbf{u}_{act})
$$

(2.153)
can be used to evaluate the bias between the solutions screw axes.

The solution evaluation can also be done if we compute $\xi = vect(log(\mathbf{X}_{act}^{\dagger} \mathbf{X}_{alg}^{\dagger})), \xi$ represents the screw dual vector that describes the rigid displacement between $\mathbf{X}_{alg}$ and $\mathbf{X}_{act}$. Based on theorem 1, $\xi$ can be decomposed in $\xi = \alpha_\xi \mathbf{u}_\xi$. The second measurement of the error between the estimated solution and the actual solution will be:

$$
\epsilon_2 = \alpha_\xi
$$

(2.154)

The last evaluation criteria will use part of the previous measurement $\epsilon_2 = \alpha_\xi + \epsilon d_\xi$ and the norm of the difference between the displacements components of $\mathbf{X}_{act}$ and $\mathbf{X}_{alg}$ [85]

$$
\epsilon_3 = \alpha_\xi + \epsilon ||\left[\mathbf{X}_{act}(4, 1) - \mathbf{X}_{alg}(4, 1); \mathbf{X}_{act}(4, 2) - \mathbf{X}_{alg}(4, 2); \mathbf{X}_{act}(4, 3) - \mathbf{X}_{alg}(4, 3)\right]||
$$

(2.155)

where $||.||$ represents the Euclidean norm.
These errors will give a complete view over the true bias of the solution generated by the algorithm when noisy data is considered in relation with the actual solution obtained from noiseless data.

The performance of the proposed algorithm is evaluated with respect to noisy data. First the cases \( n = 2 \) and \( n = 3 \) are discussed, and further the general case \( n \geq 4 \).

(i) Consider the case \( n = 2 \) with noisy data. For example \( \hat{A}_1 \) was recorded as having an angle of rotation of 3.07[rad] instead of 3[rad] (which is an approximately 4 degrees error). This would lead to the following values:

\[
\hat{A}_1 = \begin{bmatrix}
-0.9974 & -0.0715 & 0 & 0 \\
0.0715 & -0.9974 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

A similar disturbance was considered for \( \hat{B}_3 \):

\[
\hat{B}_3 = \begin{bmatrix}
0.0408 & 0.1985 & 0.9792 & -309 \\
-0.1985 & 0.9621 & -0.1868 & 59 \\
-0.9792 & -0.1868 & 0.0787 & 291 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Taking into account that the true solution (according to [68, 69]) is

\[
\begin{bmatrix}
1 & 0 & 0 & 10 \\
0 & 0.9801 & -0.1987 & 50 \\
0 & 0.1987 & 0.9801 & 100 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

we will evaluate the errors defined in the previous section (equations (103), (104), (105)).

If input data is considered \((\hat{A}_1, \hat{B}_1), (A_3, \hat{B}_3)\) results that \( \Re(e(\hat{B}_1 \times \hat{B}_3)) \neq 0 \). Algorithm 1 will generate:

\[
X_{alg}^{\dagger} = \begin{bmatrix}
1 & 0 & 0 & 14.3364 \\
0 & 0.9783 & -0.2070 & 51.0018 \\
0 & 0.2070 & 0.9783 & 112.3369 \\
0 & 0 & 0 & 1
\end{bmatrix} ; \ X_{alg}^{++} = \begin{bmatrix}
1 & 0 & 0 & 11.9888 \\
0 & 0.9801 & -0.1987 & 50.0012 \\
0 & 0.1984 & 0.9801 & 105.2789 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

The errors (103), (104) and (105) have the following values
which indicates that the output of the proposed method is very close to the true solution.

(ii) Consider the case $n = 3$ with noisy data. One of the homogeneous matrices contains noise (e.g., we will use $\hat{A}_1$ instead of $A_1$). For this case the condition $Re < b_1, b_2, b_3 > \neq 0$ is true but the rigidity constraints are not fulfilled. Thus, the solution is built and then is filtered with both techniques. In the end the filtered solution is:

$$
X^\dagger_{alg} = \begin{bmatrix}
1 & 0 & 0 & 10.0483 \\
0 & 0.9801 & -0.1987 & 50.0155 \\
0 & 0.1987 & 0.9801 & 100.0929 \\
0 & 0 & 0 & 1
\end{bmatrix} \quad ; \quad X^\dagger_{alg} = \begin{bmatrix}
1 & 0 & 0 & 10.0155 \\
0 & 0.9801 & -0.1987 & 50.0065 \\
0 & 0.1987 & 0.9801 & 100.0541 \\
0 & 0 & 0 & 1
\end{bmatrix}
$$

The errors (103), (104) and (105) have the following values:

<table>
<thead>
<tr>
<th>Solution</th>
<th>$\epsilon_1 = \text{[rad]} + \epsilon [\text{mm}]$</th>
<th>$\epsilon_2 = \text{[rad]} + \epsilon [\text{mm}]$</th>
<th>$\epsilon_3 = \text{[rad]} + \epsilon [\text{mm}]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X^\dagger_{alg}$</td>
<td>$\epsilon_1 = 0 + \epsilon 0.53603$</td>
<td>$\epsilon_2 = 0 + \epsilon 0.10846$</td>
<td>$\epsilon_3 = 0 + \epsilon 0.10588$</td>
</tr>
<tr>
<td>$X^\dagger_{alg}$</td>
<td>$\epsilon_1 = 0 + \epsilon 0.3041$</td>
<td>$\epsilon_2 = 0 + \epsilon 0.05601$</td>
<td>$\epsilon_2 = 0 + \epsilon 0.05570$</td>
</tr>
</tbody>
</table>

which indicates that the output of the proposed method is very close to the true solution.

<p>| Table 2.1 Performance evaluation for $X^\dagger_{alg}$ solutions |
|---------------------------------|---------------------------------|---------------------------------|</p>
<table>
<thead>
<tr>
<th>no of pairs</th>
<th>$\epsilon_1 = \text{[rad]} + \epsilon [\text{mm}]$</th>
<th>$\epsilon_2 = \text{[rad]} + \epsilon [\text{mm}]$</th>
<th>$\epsilon_3 = \text{[rad]} + \epsilon [\text{mm}]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>$\epsilon_1 = 0.0673 - \epsilon 12.4102$</td>
<td>$\epsilon_2 = 0.0434 - \epsilon 0.2515$</td>
<td>$\epsilon_3 = 0.0434 + \epsilon 1.6031$</td>
</tr>
<tr>
<td>20</td>
<td>$\epsilon_1 = 0.0541 - \epsilon 7.9878$</td>
<td>$\epsilon_2 = 0.0318 - \epsilon 0.0092$</td>
<td>$\epsilon_3 = 0.0318 + \epsilon 1.1872$</td>
</tr>
<tr>
<td>30</td>
<td>$\epsilon_1 = 0.0357 - \epsilon 5.6224$</td>
<td>$\epsilon_2 = 0.0172 + \epsilon 0.3486$</td>
<td>$\epsilon_3 = 0.0172 + \epsilon 1.0827$</td>
</tr>
<tr>
<td>40</td>
<td>$\epsilon_1 = 0.0306 - \epsilon 5.0700$</td>
<td>$\epsilon_2 = 0.0155 + \epsilon 0.2595$</td>
<td>$\epsilon_3 = 0.0155 + \epsilon 0.8691$</td>
</tr>
<tr>
<td>50</td>
<td>$\epsilon_1 = 0.0406 - \epsilon 7.7867$</td>
<td>$\epsilon_2 = 0.0191 + \epsilon 0.2501$</td>
<td>$\epsilon_3 = 0.0191 + \epsilon 0.8974$</td>
</tr>
<tr>
<td>60</td>
<td>$\epsilon_1 = 0.0241 - \epsilon 4.3471$</td>
<td>$\epsilon_2 = 0.0130 + \epsilon 0.0447$</td>
<td>$\epsilon_3 = 0.0130 + \epsilon 0.7102$</td>
</tr>
<tr>
<td>70</td>
<td>$\epsilon_1 = 0.0269 - \epsilon 4.0714$</td>
<td>$\epsilon_2 = 0.0119 + \epsilon 0.2878$</td>
<td>$\epsilon_3 = 0.0119 + \epsilon 0.7034$</td>
</tr>
<tr>
<td>80</td>
<td>$\epsilon_1 = 0.1023 - \epsilon 4.4792$</td>
<td>$\epsilon_2 = 0.0112 + \epsilon 0.3096$</td>
<td>$\epsilon_3 = 0.0112 + \epsilon 0.6504$</td>
</tr>
<tr>
<td>90</td>
<td>$\epsilon_1 = 0.0211 - \epsilon 3.2962$</td>
<td>$\epsilon_2 = 0.0101 + \epsilon 0.0365$</td>
<td>$\epsilon_3 = 0.0101 + \epsilon 0.5885$</td>
</tr>
<tr>
<td>100</td>
<td>$\epsilon_1 = 0.0254 - \epsilon 4.6949$</td>
<td>$\epsilon_2 = 0.0144 - \epsilon 0.0359$</td>
<td>$\epsilon_3 = 0.0144 + \epsilon 0.6017$</td>
</tr>
</tbody>
</table>
2.3 Applications

Table 2.2 Performance evaluation for $X_{alg}^{++}$ solutions

<table>
<thead>
<tr>
<th>no of pairs</th>
<th>$e_1 = [rad] + \varepsilon [mm]$</th>
<th>$e_2 = [rad] + \varepsilon [mm]$</th>
<th>$e_3 = [rad] + \varepsilon [mm]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>$e_1 = 0.1021 - \varepsilon 17.7416$</td>
<td>$e_2 = 0.0459 + \varepsilon 0.3476$</td>
<td>$e_3 = 0.0459 + \varepsilon 1.8244$</td>
</tr>
<tr>
<td>20</td>
<td>$e_1 = 0.0548 - \varepsilon 10.0615$</td>
<td>$e_2 = 0.0248 + \varepsilon 0.2350$</td>
<td>$e_3 = 0.0248 + \varepsilon 1.4370$</td>
</tr>
<tr>
<td>30</td>
<td>$e_1 = 0.0552 - \varepsilon 10.3325$</td>
<td>$e_2 = 0.0235 + \varepsilon 0.0157$</td>
<td>$e_3 = 0.0235 + \varepsilon 1.1908$</td>
</tr>
<tr>
<td>40</td>
<td>$e_1 = 0.0375 - \varepsilon 6.0916$</td>
<td>$e_2 = 0.0175 + \varepsilon 0.3958$</td>
<td>$e_3 = 0.0175 + \varepsilon 1.0492$</td>
</tr>
<tr>
<td>50</td>
<td>$e_1 = 0.0359 - \varepsilon 6.0990$</td>
<td>$e_2 = 0.0148 + \varepsilon 0.5295$</td>
<td>$e_3 = 0.0148 + \varepsilon 1.0089$</td>
</tr>
<tr>
<td>60</td>
<td>$e_1 = 0.0265 - \varepsilon 5.3081$</td>
<td>$e_2 = 0.0122 - \varepsilon 0.0430$</td>
<td>$e_3 = 0.0122 + \varepsilon 0.8917$</td>
</tr>
<tr>
<td>70</td>
<td>$e_1 = 0.0202 - \varepsilon 3.8692$</td>
<td>$e_2 = 0.0074 + \varepsilon 0.3638$</td>
<td>$e_3 = 0.0074 + \varepsilon 0.6613$</td>
</tr>
<tr>
<td>80</td>
<td>$e_1 = 0.0253 - \varepsilon 5.1217$</td>
<td>$e_2 = 0.0102 + \varepsilon 0.2074$</td>
<td>$e_3 = 0.0102 + \varepsilon 0.7093$</td>
</tr>
<tr>
<td>90</td>
<td>$e_1 = 0.0229 - \varepsilon 4.2660$</td>
<td>$e_2 = 0.0088 + \varepsilon 0.3316$</td>
<td>$e_3 = 0.0088 + \varepsilon 0.8241$</td>
</tr>
<tr>
<td>100</td>
<td>$e_1 = 0.0267 - \varepsilon 5.1295$</td>
<td>$e_2 = 0.0093 + \varepsilon 0.3386$</td>
<td>$e_3 = 0.0093 + \varepsilon 0.6491$</td>
</tr>
</tbody>
</table>

(iv) For the case $n \geq 4$ we have adopted the following protocol: perform simulations for each number of data pair $n = 10, 20, ..., 100$ and calculate the average errors (103), (104), (105) in the course of 10 trials. Consider the ground truth solution to the tensor calibration problem to be

$$X_{gt} = \begin{bmatrix} 0.9182 & -0.3961 & 0.0079 & 3.8 \\ 0.3875 & 0.9020 & 0.1902 & 14 \\ -0.0825 & -0.1715 & 0.9817 & 90 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The homogeneous matrices $B_i$, $i = 1, n$ were generated automatically. From their $A_i$, $i = 1, n$ correspondents we have recovered the logarithm and added as disturbance a normal distributed random variable of mean 0 and standard deviation 0.05 for the rotation angle, a normal distributed random variable of mean 0 and standard deviation 1 for displacement and a normal distributed random variable of mean 0 and standard deviation 0.02 for the dual axis.

The numerical evaluation is presented in Tables 1 and 2. Each table contains the values of the evaluation equations (103),(104) and (105). For both of the filtering techniques each of the three errors is decreasing when the number of pairs increases. From the behavior of $e_1, e_2, e_3$ we can conclude that Algorithm 1 is convergent towards the ground truth, thus underlying the reliability of the orthogonal tensor based solutions proposed in this paper.
Chapter 3

Path Planning for multi-robot task allocation problems

In this chapter there are presented the contributions made in the area of motion planning for multi-robot systems. First, the research context is detailed. Next, sample gathering problems and solutions are presented for both mobile robots with wheels and drones. In the end of the chapter, applications both in simulation and real-time time experiments are analyzed.

3.1 Research context

The path planning problem tackled by this work is a particular case of Capacity and Distance constrained Vehicle Routing Problem (CDVRP) [86] with capacity equal to one and the distance constrains related to the limited energy. For this problem, many algorithms have been provided, for the exact methods [87, 88] and also for heuristic methods [89–92]. However, the chosen problem for this work is different in multiple aspects. First, in the CDVRP problem, the capacity of each vehicle (robot in our case) should be greater than the demand of each vertex [86]. In our case this cannot hold since the robot capacity is one (we assume that each robot can transport maximum one good) and the number of goods at the vertices is, in many cases, greater than one. Second, the heuristics considered in this work, which include relaxing the optimal solution of the MILP to a quadratic programming problem, have not been considered for the CDVRP. Finally, most of the works on CDVRP try to characterize worst-case scenarios through cost differences between heuristic and optimal methods. In this paper we are more interested in the computational complexity and we evaluate the proposed solutions using numerical simulations.
None of the mentioned works contains a directly-applicable formulation that yields a solution for the specific problem. This work builds on solutions reported in [93–95]. In [93] we constructed a MILP problem that solves the minimum time sample gathering problem. Different than in [93], we also design a MILP solution that can be used when the initial problem is infeasible due to scarce energy limits. Besides the MILP formulations, one of the main goals of this research is to provide computationally efficient sub-optimal solutions for the targeted problems. The MILP solution was relaxed to a QP formulation in [94]. Here we also construct an iterative sub-optimal solution inspired by [95] as a QP alternative. Based on extensive simulations that involve the three proposed solutions, we conclude with a decision scheme that helps an user to choose the appropriate method for a specific problem instance.

The purpose of the research is to plan a team of mobile agents such that they gather the samples scattered throughout the environment into a storage facility. The problem’s hypothesis consists in a team of mobile robots which has to bring to a deposit region a set of samples that exist in an environment at known locations. Two type of agents are considered: mobile robots with wheels and drones. The environment is set to be an indoor wear-house and is modeled by a graph where an arc weight corresponds to consumed energy and time for moving between the linked nodes. Each robot has limited energy, and the goal is to collect all samples in minimum time. Because the robots are initially deployed in the storage (deposit) node and given the static nature of the environment, it is customary to build movement plans before the agents start to move. The problem reduces to a specific case of optimal assignment or task allocation [40, 42, 43]. The research combines results from previous work reported in [30, 94, 96].

An important advantage in using drones in indoor applications is the flexibility of the work environment. Autonomous maneuvering inside such an environment is highly challenging as most of the space is occupied with storage facilities loaded with different types of goods. Current works are focused mainly on scanning applications of the goods using fiducial markers and RFID tags [97] or shape detection [98].

In this paper we tackle a different application, one that is focused on gathering goods from indoor storage zones and their transportation into specific delivery areas, similar to trucks that should be automatically loaded with different goods from a warehouse. The entire path planning is intended to be done using optimal performances. A team of micro-aerial vehicles is used to put into practice the proposed goods gathering and delivery strategies. Transportation represents today about 5% of gross domestic product (GDP) and directly employs around 10 million people in the EU. In particular, in logistics, warehousing and storage represent up the 15% of the current costs (https://ec.europa.eu/jrc/en/research-topic/transport-sector-economic-analysis).
This work is focused on solving a type of vehicle routing problem [99, 26, 41] (VRP) specifically for drones goods gathering and deployment scenarios. The main hypotheses are: drones have predefined carrying capacities and have limited flight times/energy. Their task is to gather and deploy goods in specific storage areas in optimal manner in terms of energy consumption. To recover these paths, the constraints and objective of the problem can be modelled as a suite of Binary Integer Programming (BIP) problems [100]. On the other hand, this problem can be seen as a particular case of Capacity and Distance constrained Vehicle Routing Problem (CDVRP) [99] with distance constrains related to the limited energy. For this problem, many algorithms have been provided, for the exact methods [87, 88] and for heuristic methods [89, 90, 101, 92]. However, our problem is different since in the CDVRP problem, the capacity of each vehicle (drone in our case) should be greater than the demand of each vertex [99]. In our case this is not true in general. Second, most of the works on CDVRP try to characterize worst-case scenarios through cost differences between heuristic and optimal methods, in this work the solution yielded by our approach is validated in two phases. First a validation is carried in Matlab, underlying the theoretical properties via numerical simulation. Next, an experimental platform is used to conduct preliminary real-time implementations using a team of micro-aerial vehicles to pick and deploy samples from an indoor work space.

3.2 Sample gathering problems and solutions

3.2.1 Sample gathering using a team of mobile robots with wheels

Consider a team of $N_R$ identical robots that are labeled with elements of set $R = \{r_1, r_2, \ldots, r_{N_R}\}$. The robots “move" on a weighted graph $G = (V, E, c)$, where $V = \{v_1, v_2, \ldots, v_{|V|}\}$ is the finite number of nodes (also called locations or vertices), $E \subseteq V \times V$ is the adjacency relation corresponding to graph edges, and $c : E \rightarrow \mathbb{R}_+$ is a cost (weight) function.

We mention that there are multiple approaches for creating such finite-state abstractions of robot control capabilities in a given environment [31, 32]. A widely used idea is to partition the free space into a set of regions via cell decomposition methods, each of these regions corresponding to a node from $V$ [102, 35, 103]. The graph edges correspond to possible robot movements between adjacent partition regions, i.e., $\forall v, v' \in V$, if an agent can move from location $v$ to $v'$ without visiting any other node from graph, then $(v, v') \in E$. Each edge corresponds to a continuous feedback control law for the robot such that the desired movement is produced, and various methods exist for designing such control laws based on agent dynamics and partition types [104, 105]. Alternatives to cell decomposition methods,
as visibility graphs or generalized Voronoi diagrams, can also produce discrete abstractions in form of graphs or transition systems [31].

The graph $G$ is assumed connected, and the adjacency relation $E$ is symmetric, i.e. if a robot from $R$ can travel from location $v$ to $v'$, then it can also move from $v'$ to $v$. For any $(v,v') \in E$, we consider that the cost $c(v,v')$ represents the amount of energy spent by the robot for performing the movement from $v$ to $v'$ and that $c(v',v) = c(v,v')$. By assuming identical agents with constant velocity and a homogenous environment (i.e. the energy for following an arc is proportional with the distance between linked nodes), we denote the time necessary for performing the movement from location $v$ to $v'$ by $\gamma \cdot c(v,v')$, where $\gamma \in \mathbb{R}_+$ is a fixed value.

Initially, all agents are deployed in a storage (deposit) node $v_{|V|}$ (labeled for simplicity as the last node in graph $G$), and each robot $r \in R$ has a limited amount of energy, given by map $\mathcal{E} : R \rightarrow \mathbb{R}_+$, for performing movements on abstraction $G$. For homogenous environments and constant moving speeds, energy $\mathcal{E}(r)$ can be easily linked with battery level of robot $r$, with the distance it can travel, or the sum of costs of followed edges.

There are $N_S$ samples or valuable items scattered throughout the environment graph $G$. The samples are indexed (labeled) with elements of set $S = \{1, 2, \ldots, N_S\}$, while a map $\pi : S \rightarrow V$ shows to which node each sample belongs.

For every robot $r \in R$ find a moving strategy on $G$ such that:

- the team of robots gathers (collects) all samples from graph $G$ in the storage node $v_{|V|}$ within minimum time;
- each robot can carry at most one sample at any moment;
- the total amount of energy spent by each robot is at most its initially available energy.

The chosen problem is related to the so-called Set Partitioning Problems (SPPs) [42], which are employed in various task allocation problems for mobile agents. A SPP formulation aims to find a partition of a given set such that an utility function defined over the set of acceptable partitions with real values is maximized. Various SPPs are solved by using Operations Research formulations that employ different standard optimization problems. In this case, the given set is $S$ (samples to be collected), while the desired partition should have $N_R$ disjoint subsets of $S$, each subset corresponding to the samples a robot should collect. The utility relates to the necessary time required for collecting all samples (being a maximum value over utilities of elements of obtained partition), while the partition is acceptable if each robot has enough energy to collect its samples. The maximization over individual utilities show that our problem is more complicated than standard SPPs. Since an SPP is NP-hard
we conclude that problem is also NP-hard. Therefore, we expect computationally intensive solutions for optimally solving Problem 3.2.1, while suboptimal relaxations may be used when an optimal solution does not seem tractable.

Since the sample deployment and robot energy limits are known, the searched solution is basically an off-line computed plan (sequence of nodes) for each robot such that the mission requirements are fulfilled. The first requirement from Problem 3.2.1 can be regarded as a global target for the whole team (properly assign robots to collect samples such that the overall time for accomplishing the task is minimized), while the last two requirements are related to robot capabilities. As in many robot planning approaches such that global tasks are accomplished, we do not account for inter-robot collisions when developing movement plans. In real applications, such collisions can be avoided by using local rules during the actual movement, and the time (or energy) offset induced by such rules is negligible with respect to the total movement time (or required energy). Clearly, in some cases Problem 3.2.1 may not have a solution due to insufficient available energy for robots, these situations will be discussed during solution description.

For supporting the problem formulation and solution development, we introduce an example that will be discussed throughout the next sections. Thus, we assume an environment abstracted to the graph from Fig. 3.1, composed by 10 nodes \(v_1, v_2, \ldots, v_{10}\), with the deposit \(v_{10}\). The costs for moving between adjacent nodes are marked on the arcs from Fig. 3.1, e.g. \(c(v_{10}, v_8) = 2\). The team consists of 3 robots labeled with elements of set \(R = \{r_1, r_2, r_3\}\). For simplicity of exposition, we consider \(\gamma = 1\) (the constant that links the moving energy with necessary time) and equal amounts of energy for robots, \(E(r) = 100, \forall r \in R\). There are 14 samples scattered in this graph (labeled with numbers from 1 to 14), with locations given by map \(\pi\), e.g. \(\pi(9) = \pi(10) = v_3\).

The problem requires a sequence of movements for each robot such that all the 14 samples are gathered into storage \(v_{10}\), each robot being able to carry one sample at any time.

**Mathematical Model and Optimal Solution**

To solve Problem 1, defined in the previous section, the proposed approach consists from the following main steps:

1. Determine optimal paths in graph \(G\) from storage node to all nodes containing samples.
2. Formulate linear constraints for correctly picking samples and for not exceeding robot’s available energy based on a given allocation of each robot to pick specific samples.
3. Create a cost function based on robot-to-sample allocation and on necessary time for gathering all samples.
4. Cast the above steps in a form suitable for applying existing optimization algorithms and thus find the desired robot-to-sample allocations.

Step (i) is accomplished by running a Dijkstra algorithm [107] on the weighted graph \( G \), with source node \( v_{|V|} \) and with multiple destination nodes: \( v \in V \) for which \( \exists s \in S \) such that \( \pi(s) = v \). Note that a single run of Dijkstra algorithm returns minimum cost paths to multiple destinations.

Let us denote with \( path(v) \) the obtained path (sequence of nodes) from deposit \( v_{|V|} \) to node \( v \in V \) and with \( \omega(v)/2 \) its total cost\(^1\). Due to symmetrical adjacency relation of \( G \), the retour path from \( v \) to storage is immediately constructed by following \( path(v) \) in inverse order. The retour path is denoted by \( path^{-1}(v) \) and it has the same cost \( \omega(v)/2 \). Thus, for collecting and bringing to storage location the sample \( s \in S \), a robot spends \( \omega(\pi(s)) \) for the round-trip given by \( path(\pi(s)), path^{-1}(\pi(s)) \).

For solving steps (ii)-(iv), let us first define a decision function as \( x : N_R \times N_S \rightarrow \{0, 1\} \) by:

\[
x(r, s) = \begin{cases} 
1, & \text{if robot } r \text{ picks sample } s \\ 
0, & \text{otherwise} \\
\end{cases}, \forall r \in R, s \in S. \tag{3.1}
\]

\(^1\)Because we will be interested in the round-trip cost between nodes \( v_{|V|} \) and \( v \), we denote the one-way cost by \( \omega(v)/2 \). Thus, the cost of the full path will be simply denoted by \( \omega(v) \).
The actual values returned by map $x$ for any pair $(r, s) \in R \times S$ give the most important part of solution to Problem 3.2.1, and these values are unknown, yet. The outcomes of $x$ will constitute the decision variables in an optimization problem that is described next.

Step (ii) is formulated as the following set of linear constraints:

\[
\begin{align*}
\sum_{r \in R} x(r, s) &= 1, \quad \forall s \in S \\
\sum_{s \in S} \left( \omega(s) \cdot x(r, s) \right) &\leq \mathcal{E}(r), \quad \forall r \in R,
\end{align*}
\]  

(3.2)

where the first set of equalities impose that exactly one robot is sent to collect each sample, while the subsequent inequalities guarantee that the energy spent by robot $r$ for collecting all its assigned samples does not exceed its available energy.

The cost function from step (iii) corresponds to the first requirement of Problem 3.2.1. It means to minimize the maximum time among all robots required for collecting the assigned samples, and it formally translates to finding outcomes of $x$ that minimize the objective function $J(x)$ from

\[
J(x) = \min_x \max_{r \in R} \left( \gamma \cdot \sum_{s \in S} \left( \omega(s) \cdot x(r, s) \right) \right). 
\]

(3.3)

The objective function from (3.3) and the constraints from (3.2) form a minimax optimization problem [108, 109]. However, decision function $x$ should take binary values. In order to use available software tools when solving for values of $x$, step (iv) transforms the minimax optimization into a MILP problem [109, 110], by adding an auxiliary variable $z \in \mathbb{R}_+$ and additional constraints that replace the max term from (3.3). This results in the following MILP optimization:

\[
\begin{align*}
\min_{x, z} & \quad z \\
\text{s.t.:} & \quad \sum_{r \in R} x(r, s) = 1, \quad \forall s \in S \\
& \quad \sum_{s \in S} \left( \omega(s) \cdot x(r, s) \right) \leq \mathcal{E}(r), \quad \forall r \in R \\
& \quad \gamma \cdot \sum_{s \in S} \left( \omega(s) \cdot x(r, s) \right) \leq z, \quad \forall r \in R \\
& \quad x(r, s) \in \{0, 1\}, \quad \forall (r, s) \in R \times S \\
& \quad z \geq 0
\end{align*}
\]

(3.4)
The MILP (3.4) can be solved by using existing software tools [111–113]. The solution is guaranteed to be globally optimal because both the feasible set defined by the linear constraints from (3.4) and the objective function (z) are convex [114, 110]. Thus, solution of (3.4) gives the optimal outcomes for \( x \) (unknown decision variables \( x(r,s) \)), as well as the time \( z \) in which the team solves Problem 3.2.1.

Note that the obtained map \( x \) indicates the samples that have to be collected by each robot, as in (3.1). However, it does not impose any order for collecting these samples. For imposing a specific sequencing, each robot is planned to collect its allocated samples in the ascending order of the necessary costs. This means that robot \( r \) first picks the sample \( s \) for which \( x(r,s) = 1 \) and \( \omega(\pi(s)) \leq \omega(\pi(s')) \), \( \forall s' \in S \) with \( x(r,s') = 1 \), and so on for the other samples. The optimum path for collecting sample \( s \) from node \( \pi(s) \) was already determined in step (i). Under the above explanations, Alg. 5 includes the steps for obtaining an optimal solution for Problem 3.2.1.

**Algorithm 5: Optimal solution**

**Input:** \( G,R,S,\mathcal{E},\pi \)

**Output:** Robot movement plans

1. Find on graph \( G \) paths \( \text{path}(v) \) and costs \( \omega(v) \) for every \( v = \pi(s), s \in S \)
2. Solve MILP optimization (3.4)
3. if solutions \( z \) and \( x \) are obtained then
   4. for \( r \in R \) do
      5. \( \text{plan}_r = \emptyset \)
      6. \( S_{r,\text{collects}} = \{ s \in S | x(r,s) = 1 \} \)
      7. Sort set \( S_{r,\text{collects}} \) in ascending order based on costs \( \omega(\pi(s)), s \in S_{r,\text{collects}} \)
      8. for \( s \in S_{r,\text{collects}} \) do
         9. Append \( \text{path}(\pi(s)) \) to plan of robot \( r, \text{plan}_r \)
         10. Insert command to collect sample \( s \) in \( \text{plan}_r \)
         11. Append \( \text{path}^{-1}(\pi(s)) \) to \( \text{plan}_r \)
         12. Insert command to deposit sample \( s \) in \( \text{plan}_r \)
   13. Return plans \( \text{plan}_r, \forall r \in R \)
4. else
   5. Problem 3.2.1 is infeasible
   6. Return

The Dijkstra algorithm (line 1 from Alg. 5) returns paths and corresponding energy for collecting each sample, e.g., \( \text{path}(\pi(1)) = \text{path}(v_1) = v_{10}, v_9, v_1 \) and \( \omega(\pi(1)) = 16 \). The MILP (3.4) was solved in about 0.7 seconds and it returned an optimal solution with \( z = 54 \) (time for fulfilling Problem 3.2.1) and allocation map \( x \). Based on robot-to-sample allocations \( x \), lines 3-13 from Alg. 5 yield the robotic plans for collecting samples from the following
nodes (the sequences of nodes and the collect/deposit commands are omitted due to their length):

Robot $r_1$ collects samples from:

$\langle v_8 \rangle, \langle v_8 \rangle, \langle v_2 \rangle, \langle v_1 \rangle$ (time: 54)

Robot $r_2$ collects samples from:

$\langle v_3 \rangle, \langle v_2 \rangle, \langle v_1 \rangle$ (time: 54)

Robot $r_3$ collects samples from:

$\langle v_8 \rangle, \langle v_5 \rangle, \langle v_3 \rangle, \langle v_2 \rangle, \langle v_1 \rangle$ (time: 52)

In the remainder of this section we focus on the situation in which the mobile robots cannot accomplish Problem 1 due to energy constraints.

Relaxing infeasible problems: If MILP (3.4) is infesable, this means that Problem 3.2.1 cannot be solved due to insufficient available energy of robots for collecting all samples. Intuitive argument. Whenever (3.4) has a non-empty feasible set (the set defined by the linear constraints), it returns an optimal solution from this set [114, 110]. The feasible set can become empty only when the first two sets of constraints and the fourth ones from (3.4) are too stringent. The third set of constraints cannot imply the emptiness of feasible set, because there is no upper bound on $z$. It results that (3.4) has no solution whenever the first, second and fourth sets of its constraints cannot simultaneously hold. The fourth constraints cannot be relaxed, and therefore only the first two sets may imply the infeasibility of (3.4). This proves the remark, since the first constraints require all samples to be collected, while the second ones impose upper bounds on consumed energy.

MILP (3.4) has a non-empty feasible when the required energy for collecting all samples is small enough, or when the available energy limits $\delta^e(r)$ are large enough. This is because the connectedness of $G$ implies that the coefficients $\omega(\pi(s))$ are finite, $\forall s \in S$. Therefore, the first two sets of constraints from (3.4) could be satisfied even by an initial solution of form $x(r,s) = 1$ for a given $r \in R$, $\forall s \in S$, and $x(r',s) = 0$ for any $r' \in R \setminus \{r\}$.

This aspect yields the idea that one can relax the first constraints from (3.4) whenever there is no solution, i.e., collect as many samples as possible with the available robot energy. Such a formulation is given by the MILP problem (3.6), which allows that some samples are not collected (inequalities in first constraints) and imposes a penalty in the cost function for each uncollected sample. Basically, for a big enough value of $W > 0$ from (3.6), any uncollected sample would increase the value of the objective function more than the decrease resulted from saved energy. The constant $W$ can be lower-bounded by:

$$W > \gamma \cdot \sum_{s \in S} \omega(\pi(s)).$$
For this lower bound, the cost function increases whenever a sample \( s \) is not collected, because the term \( z \) decreases with \( \gamma \cdot \omega(\pi(s)) \) and the second term increases with more than this value. Since MILP optimization returns a global optimum, minimizing the cost function under constraints from (3.6) guarantees that the largest possible number of samples are collected while minimizing the necessary time.

Observe that, when (3.6) is employed, the returned value of the minimized function does not represent the time for collecting all samples, but this time is given by the returned \( z \).

\[
\begin{align*}
\min_{x,z} & \quad z - W \cdot \sum_{s \in S} \sum_{r \in R} x(r,s) \\
\text{s.t.:} & \quad \sum_{r \in R} x(r,s) \leq 1, \ \forall s \in S \\
& \quad \sum_{s \in S} \left( \omega(\pi(s)) \cdot x(r,s) \right) \leq \varepsilon(r), \ \forall r \in R \\
& \quad \gamma \cdot \sum_{s \in S} \left( \omega(\pi(s)) \cdot x(r,s) \right) \leq z, \ \forall r \in R \\
& \quad x(r,s) \in \{0, 1\}, \ \forall (r,s) \in R \times S \\
& \quad z \geq 0
\end{align*}
\]

3.2.2 Sample gathering using a team of quadcopters

Problem Statement

In an indoor environment (for example in a warehouse) we assume the existence of a finite number \(|S|\) of storages \( S = \{s_1, s_2, \ldots, s_{|S|}\} \) and the same number of \(|S|\) drones. Each storage contains goods of same type and to each storage a drone is assigned. The drone assigned to a storage can transport only the type of goods available at the corresponding storage. Furthermore, the carry capacity and the energy of each drone are finite. The storages are distributed in the environment at different positions.

In the same indoor environment there are also a finite number of delivery areas \( A = \{a_1, a_2, \ldots, a_{|A|}\} \). Each \( a_i \in A \) requires goods from the storages, for example delivery area \( a_1 \) requires \( n_1 \) goods from \( s_1 \), \( n_2 \) goods from \( s_2 \), \ldots, \( n_{|S|} \) goods from \( S_{|S|} \).

When the drone battery is low or it needs to be loaded with goods, it can be recharged at the assigned storage (in parallel with their loading with goods). The problem that we want to solve is to compute the sequences for gathering and delivering of goods by each drone. We also assume that all deliveries should be performed in an imposed time limit \( T \).
3.2 Sample gathering problems and solutions

**Formal solution**

The constraints and objective of our problem can be casted as a suite of $|S|$ Binary Integer Programming (BIP) problems \[100\], which can be solved by using existing optimization routines as \[115, 111, 116\]. The below BIP formulation is inspired by Mixed Integer Linear Programming (MILP) descriptions from \[117–119\], where different vehicle routing problems were solved by multiple movement tours.

Note that we can individually find a trajectory for each drone, rather than trying to find at once trajectories for all team members. This is because a type of goods can be handled only by one drone. Thus, we can iterate our solution for all drones in order to find each individual trajectories. After that, the only potential conflict can arise if two or more drones intersect along trajectories or at the same delivery area. Such potential conflicts can be solved by reordering tours of some drones, or by small delays when delivering goods, but such algorithms are not focused by this work.

For a specific drone and storage $s \in S$, let us use the following notations (the drone’s (and storage’s) ID is not included as index, for simplifying the formulations):

- The carrying capacity of the drone (maximum number of simultaneously transported goods) is $c$, and its available energy between successive visits at storage area is $E$.

- The storage (charging) area is denoted by “0”, and the goods to be delivered by the robot are denoted by positive integers from the set $G = 1, 2, \ldots, |G|$. $|G|$ is the number of goods that should be transported to delivery areas, while $G$ is containing the indices of the delivery areas where each good should be transported. For example, $G = \{1, 1, 3\}$ means that two goods should be transported to delivery area $a_1$ and 1 good should be transported to delivery area $a_3$. We know positions of storage area and delivery points from set $G$, and the drone moves (at a specific height) on straight lines connecting such positions. Thus, its motion can be regarded as corresponding to arcs in a complete graph with set of nodes (positions) denoted by $P = \{0\} \cup G$.

- The distance to be traveled between nodes $i$ and $j$ is denoted with $d(i, j), \forall i, j, \in P$. If $i$ and $j$ are at the same position (i.e., two goods to be delivered at the same point), then $d(i, j) = 0$ (the drone does not have to move between deliveries of $i$ and $j$).

- The energy spent for moving between positions $i$ and $j$ is denoted by $m^e \cdot d(i, j)$, and the corresponding time is $m^\tau \cdot d(i, j)$. Thus, $m^e$ and $m^\tau$ are coefficients giving the energy and time necessary to move on a unitary distance. Of course, if one desires to handle nonlinear dependencies of energy/time versus distance, the products $m^e / |\tau| \cdot d(i, j)$ can be replaced by other weighting values on each graph arc.
• For delivering (unloading) a good, the drone spends $ue$ energy and $u^e$ time.

• When in storage area, it takes $f^\tau$ time for filling the drone with goods (up to capacity $c$) and charging it. The drone does not consume energy in storage position.

• We assume that drone performs $N_t$ tours for accomplishing all deliveries. A tour is defined as the motion between two successive visits on the storage (charging) area, e.g., in a tour the drone can leave the storage node “0”, deliver up to $c$ goods, and then return to the charging position. Some tours can be empty, meaning that nothing is delivered in that tour, and the drone remains in storage area. We initially set the value of $N_t$ to $N_t = ceil\left(\frac{|G|}{c}\right)$, where “ceil” denotes rounding towards infinity. This value is set under the hypothesis that the drone has enough energy to deliver $c$ goods without charging, and further changes of $N_t$ will be later mentioned. For each tour $t$ we define a binary variable $z^t$ as:

$$z^t = \begin{cases} 
1, & \text{if drone delivers at least one good in tour } t \\
0, & \text{otherwise}
\end{cases}$$  (3.7)

• The delivery of a good $g \in G$ in a tour $t$ is captured by the binary variable $x^t_g$, defined as:

$$x^t_g = \begin{cases} 
1, & \text{good } g \text{ is delivered in tour } t \\
0, & \text{otherwise}
\end{cases}$$  (3.8)

• The possible movements between positions $i, j \in P, i \neq j$ in a tour $t$ are captured by the binary variables $y^t_{i,j}$, as follows:

$$y^t_{i,j} = \begin{cases} 
1, & \text{if in tour } t \text{ the drone moves directly from position } i \text{ to position } j \\
0, & \text{otherwise}
\end{cases}$$  (3.9)

Based on the above notations, the movement and delivery strategy for the drone is completely known if one gives values to all binary variables $x^t_g, y^t_{i,j}, z^t, \forall g \in G, i, j \in P (i \neq j), t \in \{1, 2, \ldots, N_t\}$. That is, we have to find specific values to these $N_t \cdot (|G| + |P|^2 - |P| + 1)$ binary variables, such that the drone accomplishes its task while optimizing a cost function. The cost function is chosen to be the total consumed energy by the drone, and all goods should be delivered in the upper-bound time limit $T$. Thus, we formulate the BIP problem (3.10) with the unknowns $x^t_g, y^t_{i,j}, z^t$. 
3.2 Sample gathering problems and solutions

\[(i) \min_{x^t_g, y^t_{i,j}, z^t} \sum_{t \in \{1,2,\ldots,N_t\}} \left( \sum_{g \in G} u^t \cdot x^t_g + \sum_{i,j \in P, i \neq j} m^t \cdot d(i,j) \cdot y^t_{i,j} \right) \]

subject to:

\[(ii) \sum_{g \in G} u^t \cdot x^t_g + \sum_{i,j \in P, i \neq j} m^t \cdot d(i,j) \cdot y^t_{i,j} \leq E, \forall t \in \{1,2,\ldots,N_t\} \]

\[(iii) \sum_{g \in G} \left( \sum_{g \in G} u^t \cdot x^t_g + \sum_{i,j \in P, i \neq j} m^t \cdot d(i,j) \cdot y^t_{i,j} + f^t \cdot z^t \right) \leq T \]

\[(iv) \sum_{g \in G} x^t_g \leq c, \forall t \in \{1,2,\ldots,N_t\} \]

\[(v) \sum_{g \in G} x^t_g = 1, \forall g \in G \]

\[(vi) \sum_{g \in G} x^t_g \geq z^t, \forall t \in \{1,2,\ldots,N_t\} \]

\[(vii) \sum_{g \in G} x^t_g \leq c \cdot z^t, \forall t \in \{1,2,\ldots,N_t\} \]

\[(viii) \sum_{i \in P, i \neq g} y^t_{i,g} = x^t_g, \forall t \in \{1,2,\ldots,N_t\}, \forall g \in G \]

\[(ix) \sum_{j \in P, j \neq g} y^t_{g,j} = x^t_g, \forall t \in \{1,2,\ldots,N_t\}, \forall g \in G \]

\[(x) \sum_{g \in G} y^t_{0,g} = z^t, \forall t \in \{1,2,\ldots,N_t\} \]

\[(xi) \sum_{g \in G} y^t_{g,0} = z^t, \forall t \in \{1,2,\ldots,N_t\} \]

\[(xii) \sum_{\alpha \in P, M} \sum_{\beta \in M} y^t_{\alpha,\beta} \geq x^t_g, \forall t \in \{1,2,\ldots,N_t\}, \forall M \subset G, \forall g \in M \]

\[(xiii) x^t_g, y^t_{i,j}, z^t \in \{0,1\}, \forall g \in G, \forall i, j \in P (i \neq j), \forall t \in \{1,2,\ldots,N_t\} \]
The cost function and constraints of BIP (3.10) correspond to the problem requirements, as follows:

(i) The cost function is the total energy spent by the drone for delivering goods and for moving between positions from $P$ over all tours. Recall that $x$ and $y$ are binary variables (definitions (3.8) and (3.9)), so their zero values do not alter the cost function (no good delivered or no movement performed), while their one value adds the energy consumed for one of these movement/delivery actions to the cost function.

(ii) The energy spent in each tour $t$ should not exceed the imposed bound $E$. The left-hand side sums correspond to delivery energy and to moving energy, respectively.

(iii) The total time should be less or equal to the imposed time bound $T$. Here, time coefficients are used, including the filling time, with the same idea as for the cost function, since $x$, $y$ and $z$ unknowns are binary.

(iv) In each tour $t$, the total number of delivered goods cannot exceed the drone carrying capacity $c$.

(v) Each good from set $G$ will be eventually delivered.

(vi), (vii) For each tour, the value of $z^t$ (according to (3.7)) is enforced here. If no good is to be delivered in tour $t$ (all $x^t_g$ are 0) then $z^t$ becomes 0 due to constraint (vi). Contrary, if at least one good is delivered (and up to $c$ goods) then $z^t$ becomes 1 due to constraint (vii).

(viii), (ix) If good $g$ is to be delivered in tour $t$ (i.e., $x^t_g$ is 1), the drone should arrive to the corresponding delivery position (constraint (viii) enforces that one of possible incoming movements $y^t_{i,g}$ is performed). In the above case, the drone should also leave the delivery position after dropping $g$ (constraint (ix) enforces departure from position $g$).

(x), (xi) If in tour $t$ at least one delivery is to be made ($z^t$ is 1), the storage area (position “0”) should be left (according to constraint (x)), while at the end of tour $t$ a movement back to the storage area is performed (according to constraint (xi)). Note the similarity of these constraints with enforcing values of $y$ based on values of $x$ as in constraints (viii), (ix).

(xii) Such inequalities are generally used in TSP problems under the name of sub-tour elimination [101, 120, 118, 119], and here we tailor them for each tour $t$. Without such constraints, it is possible to have values for $x$, $y$, $z$ that satisfy all the other restrictions,
but the movement of the drone (given by $y_{i,j}$) is not possible, since it would consist in disjoint sequences (sub-tours). To avoid this, set $M$ spans the power set of points $P$, except storage position “0” and set of goods $G$ (already handled in constraints (x)-(xi)) and enforces that if a good is to be delivered ($x'_g$ is 1, $g \in M$), the drone should arrive in set $S$ from positions outside of $M$. Thus, for performed deliveries, each set $M$ is connected with other points from $P$, fact that avoids disjoint sequences in the same tour.

(xiii) The BIP (3.10) has a number of $N_t \cdot (|G|^2 + 2|G| + 1)$ binary variables (by replacing $|P| = |G| + 1$ in the before-mentioned number of variables).

For implementing and solving BIP (3.10), all unknowns $x, y, z$ are arranged in a vector $X$ and all equalities and inequalities are written in form $A_{eq} \cdot X = b_{eq}$, respectively $A_{ineq} \cdot X \leq b_{ineq}$. After this, we employ any specific BIP-solving software tool, as [115, 111, 116], and obtain the optimal solution given by specific values of $x, y, z$. Having these values, the actions of the drone robots are uniquely obtained. Thus, in each tour, its movements are given by $y'_{i,j}$ which are 1 (see (3.9)). Each tour when drone moves (i.e., for which $z'$ is 1) starts from and returns to position “0”. The delivery actions are triggered based on values 1 for $x'_g$, as stated by (3.8).

If BIP (3.10) returns no feasible solution, this can be due to two scenarios. On one hand, the energy limit $E$ may be too restrictive for the current number of tours, case in which $N_t$ should be increased and the problem run again. On the other hand, the problem hypothesis may be unfeasible, e.g., given positions $P$ and moving/delivery/charging delays $m^\tau, u^\tau, f^\tau$, it is not possible to deliver all the goods in time limit $T$. In such cases, $T$ should be increased. Alternatively, the warehouse topology can be changed in having more storage areas for the same type of goods and more drones that handle them, but in this case future research can target the optimum allocation of subsets of goods to available drones.

## 3.3 Applications

### 3.3.1 Path planning for a team of mobile robots with wheels

#### Sub-Optimal Planning Methods

In the general case, a MILP optimization is NP-hard [121]. The computational complexity increases with the number of integer variables and with the number of constraints, but exact complexity orders or upper-bounds on computational time cannot be formulated [117]. These notes imply that for some cases the complexity of MILP (3.4) or (3.6) may render the solution
as being computationally intractable, although the optimization is run off-line, i.e. before robot movement.

This section includes two approaches for overcoming this issue. A reformulation of the MILP problem (3.4) as in [94] and the design of a Quadratic Programming (QP) formulation. Also, an Iterative Heuristic (IH) algorithm is inspired by allocation ideas from [95].

**Quadratic Programming relaxation**

We aim to relax the binary constraints from MILP (3.4), and for accomplishing this we embed them into a new objective function. The idea starts from various penalty formulations defined in [122], some being related to the so called big M method [114].

Let us replace the binary constraints \( x(r,s) \in \{0,1\} \) from (3.4) with lower and upper bounds of 0 and respectively 1 for outcomes of map \( x \). At the same time, add to the cost function of (3.4) a penalty term depending on \( M > 0 \), as shown in the objective

\[
\min_{x,z} z + M \cdot \sum_{r \in R} \sum_{s \in S} \left( x(r,s) \cdot (1 - x(r,s)) \right),
\]

(3.11)

For a large enough value of the penalty parameter \( M \), the minimization of the new cost function from (3.11) tends to yield a binary value for each variable \( x(r,s) \). This is because only binary outcomes of \( x \) imply that the second term from sum (3.11) vanishes, while otherwise this term has a big value due to the large \( M \). The quadratic objective from (3.11) can be re-written in a standard form of an objective function of a QP problem, and together with the remaining constraints from (3.4) we obtain the QP formulation:

\[
\min_{x,z} \quad z + M \cdot \sum_{r \in R} \sum_{s \in S} x(r,s) - M \cdot \sum_{r \in R} \sum_{s \in S} \left( x(r,s) \right)^2 \\
\text{s.t.:} \quad \sum_{r \in R} x(r,s) = 1, \quad \forall s \in S \\
\quad \sum_{s \in S} \left( \omega(\pi(s)) \cdot x(r,s) \right) \leq \varepsilon(r), \quad \forall r \in R \\
\quad \gamma \cdot \sum_{s \in S} \left( \omega(\pi(s)) \cdot x(r,s) \right) \leq z, \quad \forall r \in R \\
\quad 0 \leq x(r,s) \leq 1, \quad \forall (r,s) \in R \times S \\
\quad z \geq 0
\]

(3.12)

Under the above informal explanations and based on formal proofs from [122], the MILP (3.4) and QP (3.12) have the same global minimum for a sufficiently large value of parameter \( M^2 \).\footnote{The actual value of \( M \) is usually chosen based on numerical ranges of other data from the optimization problem, as values returned by maps \( \omega \) and \( \varepsilon \).}
3.3 Applications

[Sub-optimality or failure] Note that the cost function from (3.12) is non-convex, because of the negative term in $x(r,s)^2$. Therefore, optimization (3.12) could return local minima, while in some cases the obtained values of $x$ may even be non-binary. If the obtained outcomes of $x$ are binary, the value of completion time $z$ for collecting all samples is directly returned as the cost of QP (3.12), while otherwise a large cost is obtained due to the non-zero term in $M$ from (3.11).

The QP optimization (3.12) can be solved with existing software tools [113, 111]. As noted, it may return a sub-optimal solution. Nevertheless, such a sub-optimal solution is preferable when the optimal solution is computationally intractable. If the QP returns a (local minimum) solution with non-integer values for outcomes of map $x$, then this result cannot be used for solving Problem 3.2.1.

Consider again the previous example. The paths in $G$ and outcome values of map $ω$ were already computed, where the optimal cost from MILP (3.4) was 54. QP (3.12) was solved in less than 0.4 seconds and it lead to a sub-optimal total time of 60 for bringing all samples in $v_{10}$. The robots were allocated to collect samples as follows:

Robot $r_1$ collects samples from:

$$(v_8), (v_3), (v_2), (v_1) \quad \text{time : 60}$$

Robot $r_2$ collects samples from:

$$(v_8), (v_8), (v_3), (v_2), (v_1) \quad \text{time : 48}$$

Robot $r_3$ collects samples from:

$$(v_5), (v_2), (v_2), (v_1) \quad \text{time : 52}$$

A similar QP relaxation may be constructed for MILP (3.6) by considering $M >> W$.

Iterative solution

This subsection proposes an alternative sub-optimal allocation method, described in Alg. 6. The method iteratively picks an uncollected sample whose transport to deposit requires minimum energy (line 6) and assigns it to a robot that has spent less energy (time) than other agents (lines 7-14). If a robot does not have enough energy to pick the current sample $s$, it is removed from further assignments (lines 15-16), because the remaining samples would require more energy than $ω(π(s))$. If the current sample $s$ cannot be allocated to any robot, the procedure is stopped (lines 17-18), and in this case some samples remain uncollected. When Alg. 6 reaches line 19, the robot assignments constitute a solution to Problem 3.2.1 for collecting all samples from $S$. The robot-to-sample allocations returned by Alg. 6 can be easily transformed to robot plans, as in lines 4-13 from Alg. 5. The total time for completing the mission can be easily computed by maximizing over the times spent by each robot.
Algorithm 6: Iterative heuristic solution

**Input:** $R, S, w, E$

**Output:** Robot-to-sample assignments

1. $R_{assign} = R$
2. $S_{uncollected} = S$
3. Set $x(r, s) = 0, \forall (r, s) \in R \times S$
4. Let $E_{consumed}(r) = 0, \forall r \in R$
5. while $S_{uncollected} \neq \emptyset$ do
   6. $S_0 = S_{uncollected}$
   7. Pick $s \in S_{uncollected}$ s.t. $\omega(\pi(s)) = \min_{s \in S_{uncollected}} \omega(\pi(s))$
   8. Sort $R_{assign}$ based on ascending order of consumed robot energy ($E_{consumed}$)
   9. for $r \in R_{assign}$ do
      10. if $w(s) \leq E(r)$ then
          11. $x(r, s) = 1$ (assign sample $s$ to robot $r$)
          12. $E(r) := E(r) - w(s)$
          13. $E_{consumed}(r) := E_{consumed}(r) + w(s)$
          14. $S_{uncollected} := S_{uncollected} \setminus \{s\}$
          15. Break “for” loop
      16. else
          17. $R_{assign} := R_{assign} \setminus \{r\}$
   18. if $S_{uncollected} = S_0$ then
      19. Return current robot-to-sample allocations $x$
20. Return robot-to-sample allocations $x$
Under these ideas, Alg. 6 can be seen as a greedy approach (first collect samples that require less energy/time), while the allocations to robots with less spent energy tries to reduce the overall time until the samples are collected.

Observe that this IH solution always returns a solution for collecting some (if not all) samples, whereas MILP may become computationally impracticable, while QP (3.12) may fail in providing a solution (Rem. 3.3.1). Moreover, the software implementation of Alg. 6 does not require additional tools, in contrast with specific optimization packages needed by MILP and QP solutions.

For illustrating Alg. 6 on the example considered in the previous sections, we give here the sample picking costs: \( \omega(\pi(s)) = 16, s = 1, \ldots, 4, \omega(\pi(s)) = 14, s = 5, \ldots, 8, \omega(\pi(s)) = 10, s = 9, 10, \omega(\pi(11)) = 8, \omega(\pi(s)) = 4, s = 12, \ldots, 14. \) First, IH solution allocates sample 12 to \( r_1 \), than 13 and 14 to \( r_2, r_3 \), sample 11 to \( r_1 \) and so on. Alg. 5 was run in 0.015 seconds and it yielded the following robotic plans:

- Robot \( r_1 \) collects samples from: 
  \( (v_8), (v_5), (v_2), (v_3), (v_1) \) (time : 56)

- Robot \( r_2 \) collects samples from: 
  \( (v_8), (v_3), (v_2), (v_1) \) (time : 60)

- Robot \( r_3 \) collects samples from: 
  \( (v_8), (v_3), (v_2), (v_1) \) (time : 44)

Additional examples

**Example A:** Let us add one more sample in node \( v_1 \) of the environment from Fig. 3.1, leading to a total number of 15 samples. By running the MILP optimization (3.4), a solution was obtained in almost 40 seconds and it leads to an optimum time of 60. The QP relaxation (3.12) was run in almost 0.4 seconds, and it implied a time cost of 62 for collecting all samples. The IH solution from Alg. 6 yielded a cost of 60 in 0.016 seconds. The actual robotic plans are omitted for this case.

**Example B:** If we assume a team of 4 robots for Example A, the MILP running time exhibits a significant decrease, being solved in less than 0.1 seconds. The QP optimization was solved in 0.5 seconds, and the IH in less than 0.02 seconds. The resulted time costs were 44 for MILP (optimum) and 50 for QP and IH (sub-optimum).

**Example C:** By adding one more robot to the team from Example B, the MILP optimization did not finish in 1 hour, so it can be declared computationally unfeasible for this situation. The QP optimization finished in slightly more that 0.5 seconds, while the IH in around 0.02 seconds. Both QP and IH solutions returned a cost of 40.
Similar modifications of the above examples suggested the following empirical ideas:

- The running time of the MILP optimization may exhibit unpredictable behaviors with respect to the team size and to the number and position of samples, leading to impossibility of obtaining a solution in some cases;

- In contrast to MILP, the running times of the QP and IH solutions have insignificant variations when small changes are made in the environment;

- When MILP optimization finishes, the sub-optimal costs obtained are generally acceptable when compared to the optimal cost;

- In some cases the QP cost was better than the one obtained by IH, while in other cases the vice versa, but again the observed differences were fairly small.

The above ideas were formulated only based on a few variations of the same example. However, they motivate the more extensive comparison performed in the next subsections between the computation feasibility and outcomes of MILP, QP and IH solutions.

As mentioned, complexity orders of MILP and QP solutions cannot be formally given. However, as customary in some studies, we here recall the number of unknowns and constraints of these optimizations. MILP (3.4) and QP (3.12) have each \( N_R \times N_S + 1 \) unknowns (from which \( N_R \times N_S \) are binary in case of MILP (3.4)) and \( 2N_R + N_S \) linear constraints. IH solution has complexity order \( O(N_R \times N_S) \), based on iterative loops from Alg. 6 but, in all our studies the execution time of the algorithm is very small.

**Real-time example:** Sample collecting experiments were implemented on a test-bed platform by using two Khepera robots equipped with plows for collecting items [123]. For exemplification, a movie is available at https://www.youtube.com/watch?v=2BQiWvquP7w. In the mentioned scenario, the graph environment is obtained from a cell decomposition [31, 102] and a greedy method is employed for planning the robots. Although the collision avoidance problem is not treated in this paper, in the mentioned experiment the possible collisions are avoided by pausing the motion of one robot. In future work we intend to embed formal tools inspired by resource allocation techniques for collision and deadlock avoidance [124, 125].

All the simulations to be presented were implemented in Matlab [113] and were performed on a computer with Intel i7 quad-core processor and 8 GB RAM.

The numerical experiments were run for almost 15 days, and they were organized by considering the following aspects:
3.3 Applications

![MILP success rate](image1)

![QP success rate](image2)

Fig. 3.2 Success rates of MILP (a) and QP (b) vs. number of robots $N_R$ and number of samples $N_S$.

(i) Time complexity orders cannot be a priori given for MILP or for non-convex QP optimization problems. Thus, the time for obtaining a planning solution solving Problem 3.2.1 is to be recorded as an important comparison criterion.

(ii) The complexity of MILP, QP and IH solutions does not directly depend on the size of environment graph $G$, except for the initial computation of map $\omega$ that further embeds the necessary information from the environment structure. Therefore, the running time of either solution is influenced by two parameters: the number of robots ($N_R$) and the number of samples ($N_S$).

(iii) Based on item (ii), we consider variation ranges $(N_R, N_S) \in [2, \ldots, 10] \times [2, \ldots, 50]$, with unit increment steps for $N_R$ and $N_S$. Note that the cases of 1 robot and/or 1 sample are trivial, and therefore are not included in the above parameter intervals.

(iv) In order to obtain reliable results for item (i), for each pair $(N_R, N_S)$ we have run a set of 50 trials. For each trial we generated a random distribution of samples in a 50 node graph. To maintain focus on time complexity, we assumed sufficiently large amounts of robot energy $\mathcal{E}$, such that Problem 3.2.1 is not infeasible due to these limitations.

(v) For each trial, the MILP optimization was deemed *failed* whenever (3.4) did not return a solution in less than 1 minute. This is because in multiple situations we observed that if no solution is obtained in less than 30-40 seconds, then the MILP (3.4) does not finish even after 2-3 hours.

(vi) For each trial from item (iv), the QP solution was deemed *failed* whenever it yielded non-binary outcomes of map $x$ (see Rem. 3.3.1). The IH solution is always *successful*. 
(vii) For each proposed solution, for each pair \((N_R, N_S)\) from (iii) and based on trials from (iv), we computed the following comparison criteria:

- **success rate**, showing the percentage of trials when the solution succeeded in outputting feasible plans;
- **computation time**, averaged over the successful trials of a given \((N_R, N_S)\) instance;
- **solution cost**, i.e. time for gathering all samples, for each successful situation.

**Results**

The results from item (vii) allow us to draw some rules that guide a user to choose MILP, QP or IH solution when solving a specific instance of Problem 3.2.1. The following figures present and comment these results, leading to the decision diagram from the end of this section.

Fig. 3.2 illustrates the success rates of the optimization problems. For a clearer understanding, Fig. 3.3 presents pairs \((N_R, N_S)\) when MILP 3.4 fails in more than 50% from each set of 50 trials. It is noted that MILP generally returns optimal solutions for small values of \(N_R\) and \(N_S\) and fails (because of optimization time limit) for larger values. QP returns usually returns feasible solutions, excepting some cases with small values of \(N_R\) and \(N_S\).

Fig. 3.4 presents the average computation time over the successful trials, for each solution we proposed. The representation is omitted for pairs \((N_R, N_S)\) when there are less than 5 (from 50) successful trials - as it is often the case for MILP solver, when \(N_R \geq 3\) and \(N_S \geq 13\). MILP time may sudden variations, whereas the times for QP and IH indicates predictable behaviors. The IH time is very small (note axis limits in Fig. 3.4) and exhibits negligible variations versus \(N_R\) and almost linear increases versus \(N_S\), due to the main iteration loop from Alg. 6.

To suggest the confidence intervals of values plotted in Fig. 3.4, we mention that:

- For \(N_R = 3\) and \(N_S = 15\), when the success rate of each optimization exceeds 98%, the standard deviations of optimization times are: 15 for MILP, 0.01 for QP, 0.002 for IH;
- For \(N_R = 10\) and \(N_S = 50\), when QP has 96% success rate, the standard deviations of optimization times are: 0.23 for QP and 0.003 for IH.

Fig. 3.5 presents the averaged relative differences of costs yielded by the three proposed solutions for solving Problem 3.2.1. As visible in Fig. 3.5(a), the QP (suboptimal) cost is usually less than 120% . . . 130% of optimal MILP cost. More specifically, from all the 25000 trials, in 8443 cases (about 33%) both optimizations succeeded. After averaging cost differences versus \((N_R, N_S)\), we obtained 348 points for representing Fig. 3.5(a), and
3.3 Applications

Fig. 3.3 MILP (3.4): 2D projection for failures, defined as success rate of less than 50%.

in 330 cases the cost difference was less than 20%. Fig. 3.5(b) compares the costs yielded by the suboptimal solutions by representing variations of the IH cost related to the QP one. It follows that usually IH yields a higher cost than QP, but the difference decreases below 5%...10% with the increase in problem complexity. Further studies can be conducted towards formulating a conjecture that gives a formal tendency for the variation of cost difference based on problem size. However, one issue for such a study is mainly given by the necessity of obtaining the optimal cost even for large problems, i.e. solving large MILP optimizations.

For more complex problems \((N_R > 10, N_S > 50)\), the time tendencies from Fig. 3.4(b),(c) and the cost differences from Fig. 3.5(b) suggest that the IH solution is preferable as a good trade-off between planning complexity and resulted cost.
Problem 1

MILP (4)

Complexity: (\(N_R, N_S\))

Energy limits (\(\varepsilon\))

\(N_R < 3\) and \(N_S < 12\)

unrestrictive

MILP (6)

\(N_R \in [3,10]\) and \(N_S \in [12,50]\)

restrictive

IH (Alg. 2)

\(N_R > 10\) or \(N_S > 50\)

restrictive

QP (8)

unrestrictive

Fig. 3.6 Decision diagram for choosing an appropriate solution for Problem 3.2.1.
3.3 Applications

3.3.2 Path Planning for a team of Autonomous Drones

Simulation validation

We consider an environment with a rectangular cuboid shape in which \(|S| = 2\) drones evolve. For easier understanding the environment, Fig. 3.7 presents a 2D projection, by denoting the number of each type of good, the storage and delivery areas. The storage areas are at the ground level and all delivery areas are at height 3 in measuring units of axes in Figure 3.7. In a tour, each drone lifts at height 3, then moves on straight lines to delivery areas, and finally moves above the storage and descents for loading. Thus, for green drone, the distance to be travelled to the left-most delivery area is:

\[
d(0, 1) = 2 \times 3 + \sqrt{5^2 + 4^2} \approx 12.
\]

![Fig. 3.7 Projection of a 3D environment, with two storage areas (green and red circles) and four delivery zones (cyan squares). The number of goods to be delivered in written on each storage area.](image)

Each drone has capacity \(c = 3\), and for simplicity of understanding the solution the time/energy coefficients \((m^r, m^e, u^r, u^e, f^r)\) are unitary. For large enough time and energy bounds, we obtain the solution from Figure 3.8, i.e., the green drone performs 2 tours, while the red one suffices 1 tour. The BIP for the green drone has 72 unknowns and 188 constraints, and it was solved with [115] in less than 0.1 seconds. The BIP for the red drone has 16 unknowns, 25 constraints and it was solved in almost 0.05 seconds. The green drone consumes almost 66 units of energy for performing all its moving and delivery actions (31.5 for tour 1, 34.3 for tour 2), the red drone consumes 35 units of energy. If the energy bound per tour \(E\) is decreased for the first drone (e.g., \(E = 30\)), the number of tours should be...
increased in order to obtain a solution from BIP (3.10), and in this case the drone would deliver at most 2 goods in each tour (although its capacity is 3).

Fig. 3.8 Sketch of movement and delivery sequences for each drone. The green drone has 2 tours (the first represented with continuous lines, and the second with dashed ones), the red drone suffices one tour.

Real-time experiments

The experimental setup (Figure 3.9) for the real time implementation of the solution proposed to the previously stated problem is built as an indoor platform supervised by an eye-in-the sky 3D-RGB sensor (Kinect v2). The platform is surrounded with safety nets which can be used for user protection during experiments.

On this platform can be identified both the goods gathering areas and the deposits. In particular, the storage areas are depicted on the floor with the ’X’ symbol of green and red colors while the delivery zones are at about 110 cm from the floor. There are three delivery zones, one green box accepting goods only from the green storage area, red box accepting goods only from red storage area and red+green box accepting goods from both delivery areas. The goods are symbolized by either green or red balls and are initially located in the corresponding storage areas. A team of two drones is used to carry out the task of gathering good and deploying them in specific containers.

The considered micro aerial vehicles are Parrot Mambo minidrones (Figure 3.10). This minidrone features the latest upgrades and performance technology that provides stability and reliable control. Altitude control is done by incorporating both an ultrasound and a
Fig. 3.9 Experimental platform

camera, the drone being able to autonomously fly at a certain height regardless of objects underneath it. Battery-wise, the Parrot Mambo has an average flight time of 10 minutes. With a control distance up to 65 meters the drone can be used in medium size indoor setups.

The experimental platform was used to test the formal solution in a real-time application. The designed setup included two storages and three areas for goods (red and green). Among the three delivery area the goods are distributed as follows: one requires only red goods, one is for green goods and one accepts both types.

Two drones are used to carry-on the task of filling the delivery containers with goods. After setting the number of goods per container, the algorithm previously presented is employed to generate the paths needed to fulfill the task. The paths will be followed by the drones while being monitored by the eye-in-the-sky camera. The visual information will be used to compute the drone attitude and feedback its position so that any deviation be nullified.

The path planning was obtained off-line using Matlab, after a scanning of the environment with the 3D-RGB sensor. This allows the storage and the goods areas position to be identified. Next, the path following was performed on-line, by running a Python script. The drones communicate with a central unit via Bluetooth, using separate threads. Drone detection is done using OpenCV library and implies identification of the front propellers (Figure 3.10) when their spinning. A threshold segmentation will generate a region that can be bounded by an ellipse which gives the drone’s orientation.
Accessing www.youtube.com/watch?v=SDDywxc6rms allows viewing of one real-time scenario. The experiment is set to gather and deploy two red goods and three green goods, considering the carrying capacity equal to two. The goals are: the green delivery receive two green goods, the red container is set to receive one red good and the mixt delivery is set to receive a red good and a green good. The task is completed as planned by the proposed algorithm and the results are illustrated in three windows: top view, side view and drone info. Regarding reliability, the team of drones follows the designed paths with low deviations, which encourages us to increase the complexity of the layout and tasks in the future experiments.
Fig. 3.11 Real-time setup. Frame from a movie of the real-time experiments. The Kinect picture in on top, the side-view on bottom, the internal drone information on left. The delivery areas are on top of the three tall shapes, and the storage areas are the two colored “X” (green and red) on the ground.
Chapter 4

Perception and control using vision based techniques

In this chapter there are presented the contributions made in the development of techniques based vision for perception and control of robotic systems. First, the contributions to perception of 3D environments using stereo vision are detailed. The second part of the chapter includes the contributions to the development of visual servoing systems.

4.1 Disparity based perception of 3D environments

Free space and obstacle detection represent two of the most active research areas that involve active or passive sensors. Both are undisputed related with an accurate ground area identification. Stereo vision systems are passive sensors that can be used to evaluate data and measure distances in front of the camera. Multiple areas, such as robotics, automotive and assistive systems, benefit from research results involving stereo vision cameras.

Any reliable stereo vision system requires to go through several phases before it can be practically exploited. A very important step is the stereo calibration phase. Once the intrinsic and extrinsic parameters are recovered, the left and right image are acquired and denoised. Next, the rectification is performed to ensure distortion removal and stereo images alignment. By this means, the correspondences can be restricted to the same line in both images and thus the computation burden in the stereo matching step can be reduced.

The disparity map refers to the displacement of the relative features or pixels between two views. Disparity maps are essential for various applications like 3D reconstruction, image based rendering, or robotic navigation. The most basic tool needed for finding corresponding points in the stereo pair is a matching cost function that measures image similarity. Most
widely used are matching cost functions that compare image intensities by their absolute or squared differences [126, 127].

Recently new computational approaches were proposed. Among them, the ELAS (Efficient Large-Scale Stereo Matching) [128] algorithm proved to have very good results regarding the density of the generated disparity map. This algorithm uses a Bayesian approach and performs quite well even in relatively low-textured images. First, the disparities of a sparse set of support points are computed using relatively strict constraints. The image coordinates of the support points are then used to create a 2D mesh via Delaunay triangulation. Finally, a prior is computed to reduce the matching ambiguities. In particular, this prior is formed by calculating a piecewise linear function induced by the support point disparities and the triangulated mesh.

In the following subsections different representations of the disparity map will be presented. For this purpose consider the disparity map to be denoted by \( D(x,y) \), where \((x,y)\) represents the position of a pixel in the image.

**V-Disparity**

The V-Disparity image [129] provides a good interpretation of the geometrical content in a scene and has a high accuracy when it comes to detecting the ground plane. This type of disparity image can be understood as the disparity histogram of each line in the disparity map. One of the most important features of the V-Disparity map is the fact that major planar surfaces in the scene have corresponding line representations in the V-Disparity image. Vertical surfaces are mapped into vertical line segments in the V-disparity image, while the ground plane corresponds to a slanted line segment.

The V-Disparity map is built based on a disparity map generated from a stereo image pair. Consider \( F \), a function attached to the input disparity image such that \( F_v(D(x,y)) = D_v(x,d) \), where \( D_v(x,d) \) is the V-Disparity map. The function \( F \) sums up all the points with the same disparity value that appear on every given row of the image.

In Algorithm 7 we present the steps that allow the computation of the V-Disparity map.

---

**Algorithm 7: V-Disparity Computation**

**Input:** Disparity map \( D(x,y) \)  
**Output:** V-Disparity Map \( D_v(x,d) \)

1. for each \textit{ith} column in \( D \) do  
   2. for each \textit{jth} line in \( D \) do  
      3. if \( D(i,j) > 0 \) then  
         4. \( D_v(j,D(i,j))++ \)
U-Disparity

The U-Disparity map [130] has the same building concept as the V-Disparity map. The main and most important difference is that the U-Disparity map is a column-wise representation of the disparity values. It provides information regarding the obstacles found in a scene, by marking them with multiple horizontal lines.

In order to build a U-Disparity map consider a function $F_u$, linked to the disparity map, such that $F_u(D(x,y)) = D_u(d,y)$, where $D_u(d,y)$ is the desired U-Disparity map. The $D_u(d,y)$ space sums up all the pixels in the initial disparity map $D(x,y)$ that have the same disparity value and are found across the same column $y$. The steps required to compute the U-Disparity map are presented in Algorithm 8.

Algorithm 8: U-Disparity Computation

<table>
<thead>
<tr>
<th>Input:</th>
<th>Disparity map $D(x,y)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>U-Disparity Map $D_v(x,d)$</td>
</tr>
</tbody>
</table>

1. for each $i^{th}$ column in $D$ do
2. for each $j^{th}$ line in $D$ do
3. if $D(i,j) > 0$ then
4. $D_v(D(i,j),i)++$

θ-Disparity

θ-disparity [131] is a recent approach for representing the 3D information of a scene. The main idea behind it is to obtain a radial representation of the significant objects in a set, with respect to a point of interest, based on the disparity map.

The first step in building the $θ$ transform is to apply a polar transform over the disparity map, where the pole of the transform is the middle pixel of the bottom line of the disparity map. This corresponds to the endpoint of the end effector in a robotics application or of the head-worn camera position in the context of assistive systems. Assuming that $D(x,y)$ has $x_{max}$ lines and $y_{max}$ columns, the coordinates of the pole will be $(x_{max}, y_{max}/2)$. The polar transformed disparity map $P(\rho, \theta)$ is based on

$$\rho \in \left\{ 0, ..., \sqrt{x_{max}^2 + \left(\frac{y_{max}}{2}\right)^2} \right\}$$

and $\theta \in \{0, ..., 180\}$
Algorithm 9: θ-Disparity Computation

**Input:** Disparity map $D(x,y)$

**Output:** θ-Disparity Map $D_\theta(\theta,d)$

1. Initialization $D_\theta \leftarrow 0$
2. Compute $P(\rho, \theta)$ polar transform of $D(x,y)$ around $(x_{max}, y_{max})$.
3. for each angle of $P$: $\theta = 0$ to $180$
4. for each disparity level $d = D_{\min}$ to $D_{\max}$
5. for each row of $P$: $\rho = 1$ to $\rho_{\max}$
6. if $P(\rho, \theta) = d$ then
7. $D_\theta(\theta,d) \leftarrow D_\theta(\theta,d) + 1$
8. $D_\theta(\theta,d) = D_\theta(\theta,d) \sin(\theta)$

The new map displays the set of disparity values that are laying along the direction angle $\theta$ in the original disparity map, $D(x,y)$ relative to a point of interest, the pole of the transform. In order to achieve the desired θ disparity map a column wise histogram will be computed. Moreover, a weighting factor $\sin(\theta)$, can be applied to each element in order to emphasise the nearby obstacles and to avoid the noticeable degeneration in the polar transformed disparity map $P(\rho, \theta)$. As we get closer to the extreme angle values (0 and 180), the values of the pixels corresponding to those columns tend to have the same value.

The θ-disparity map has only positive integer values and each point indicates the number of pixels from the initial disparity map, $D(x,y)$, that lie across a certain direction and have a disparity value $d$. The information held in the θ-disparity map can be interpreted in many different ways, depending on the goal of the application.

The steps required to obtain a theta disparity map are presented in Algorithm 9.

### 4.1.1 Obstacle detection

In this section we describe the use of the disparity representations in the previous section and detail the proposed obstacle detection algorithm. This algorithm is structured in a series of steps as depicted in Fig. 4.1.

The algorithm input is the left and right raw images acquired by the stereo vision system. Using calibration parameters, a rectification procedure is applied on the input images, the result being the rectified left and right images. Once the rectified images are available the disparity map can be computed. This task is performed using the ELAS algorithm.

The dense disparity map is further exploited using the three representations: discussed in the previous section. Each of these representations has its own contribution to the obstacle detection algorithm as detailed in the following.
4.1 Disparity based perception of 3D environments

Ground Plane Detection

Usually, regions corresponding to the ground plane match a line in the V-Disparity image. In cases where the ground plane occupies a significant region in the image, it corresponds to the most dominant line in the V-disparity map. To detect such lines in the V-Disparity map we employ an adapted version of the Hough Transform [132]. The standard Hough transform detects all the straight lines that match certain restrictions in an image. To speed up the algorithm, we use a simple thresholding algorithm on the V-Disparity image. This operation removes pixels with a low gray value that correspond to poorly represented disparities in the original image. This operation is based on the assumption that the ground plane should not have less than a user defined number of pixels located on each line.

One shortcoming associated with the standard V-disparity based ground plane detection method is that it cannot be applied on images where the ground plane is not horizontal. For non-zero roll camera rotations, estimation of the ground plane based on the V-disparity using the Hough Transform becomes a difficult task. This is due to the fact that the ground plane in the V-disparity domain is no longer represented by a single line. For larger roll angles the disparity map should be rotated before calculating the V disparity map.

To detect the roll rotation angle of the camera we can make use of stereo vision motion estimation procedures. Libviso2 [133] is a fast algorithm for computing the 6 DOF motion of a moving mono/stereo camera. For stereo sequences, libviso2, uses a procedure to extract "circular" feature matches and project feature points from the previous frame into 3D via triangulation, using the calibration parameters of the stereo camera rig. Assuming squared
pixels and zero skew, instead of minimizing the residuals in Euclidean space, the Libviso framework makes use of the intrinsic parameters of the stereo camera to minimize the residuals in the image space, where the noise level is similar for all components of the measurement vector, thus recovering the motion in a 6 parameters representation.

Once the camera motion is recovered, the roll-pitch-yaw parameterization can be easily computed and the V-disparity image can be rotated accordingly, thus ensuring robust ground plane detection results.

**Obstacle Detection using U and θ-disparity maps**

The effectiveness of the U-disparity representation is revealed in automotive applications where the detection of the road boundaries is of interest. In the U-disparity map, obstacles are usually represented as horizontal lines. This represents the main technique for obstacles identification using U-disparity map. However, when an obstacle is passing near the camera side, both the front and side faces of the obstacle are observed. The obstacle is then represented by a polyline in the V-disparity image: a horizontal part for frontage and a connected oblique part for its side face. This observation allows us to define an approach in which the U-disparity image can be easily segmented, after the ground plane is removed from the disparity map. The obstacles are represented in the image by connected-regions. Noisy pixels are removed from the V-disparity map by means of morphological operations (erosion and dilation). Each connected-region resulting after these pre-processing steps indicates a potential obstacle.

The segmentation of the θ-disparity image employs similar steps as in the processing of the U-disparity maps: the ground plane pixels are removed from the image, followed by thresholding, morphological operations and finally, detection of connected components. The strong points of the θ representation of the disparity result from the following observations: it makes no assumption about the sensor-environment geometry and it preserves the direction and angular distribution of obstacles and obstacle-free regions in a scene, while it can be estimated in a very efficient way.

The robustness of the obstacle detection algorithm is increased by performing various set operations on the obstacles pixels detected with the two methods. Depending on the actual obstacle detection application, the final segmentation results can be obtained using the intersection or the union of the results produced by the two methods. For example, in scene understanding for human assistive applications it is usually important to also produce a description of the obstacles, regarding size, shape and distance to the user. In such cases, the intersection of the two results can avoid under-segmentation and lead to better object
4.1 Disparity based perception of 3D environments

Fig. 4.2 Experiments with the KITTI dataset: Left image; Disparity map; Ground plane mask separation and tracking in consecutive frames. Moreover, a description of the free navigable space in the environment can be efficiently extracted based on the $\theta$-disparity segmentation.

**Experimental Results**

In this section we describe the experiments performed for evaluating the proposed obstacle detection framework. The evaluation is performed using data acquired from both real and virtual environments. To simulate an automotive application context for our framework, we used the KITTI Vision Benchmark Suite [134]. Images and ground truth information are generated from a virtual environment in order to test new applications for visually impaired assistive devices. The results obtained with the proposed obstacle detection framework are also made available online\(^ 1\).

*Results Obtained in a Real Environment Setup*

The KITTI benchmark dataset was captured using a station wagon dedicated to mobile robotics and autonomous driving research. The vehicle was equipped with two color and two grayscale PointGrey Flea2 video cameras (10Hz, resolution: $1392 \times 512$ pixels, opening: $90^\circ \times 35^\circ$). Camera setup is chosen such that we obtain a baseline of roughly 54 cm between the same type of cameras and that the distance between color and grayscale cameras is minimized (6cm). The scenarios are diverse, capturing real-world traffic situations and range from freeways over rural areas to inner-city scenes with many static and dynamic objects. The data from the KITTI benchmark is calibrated, synchronized and timestamped, including both rectified and raw image sequences.

The disparity maps were computed using the ELAS algorithm. As the resolution of the disparity values decreases exponentially with the distance to the camera, the disparity images were thresholded to exclude pixels with disparity values less than 25. A ground plane mask was computed using the approach described in section 4.1.1. An example image in the sequence, along with its disparity map and extracted ground plane is presented in Fig. 4.2.

The U-disparity map is computed for the disparity image after excluding the ground plane pixels. Then, morphological opening, with a square structural element of size 1, is

---

1. [https://www.youtube.com/watch?v=JWneKqtPFZs](https://www.youtube.com/watch?v=JWneKqtPFZs)
2. [https://www.youtube.com/watch?v=Wc3CeFBf3FQ](https://www.youtube.com/watch?v=Wc3CeFBf3FQ)
Perception and control using vision based techniques

Fig. 4.3 U-disparity and U-disparity labeled for an image in the KITTI sequence

Fig. 4.4 Experiments with the KITTI dataset: Polar representation of the disparity map after ground plane removal, $\theta$-disparity before and after labelling

applied to the U-disparity image after thresholding with a value of 0.05. This results in the labelled regions presented in Fig. 4.3.

The $\theta$-disparity segmentation is similarly obtained: the $\theta$-disparity image is thresholded with a value of 0.01, and morphologically opened with the same structural element (Fig. 4.4).

The results of both segmentations for the same image can be comparatively assessed in Fig. 4.5 as they are both overlaid on the initial intensity image. The obstacles pixels segmented in the $\theta$-disparity map are marked with blue, the U-disparity results are marked with red, while their intersection is marked in green.

Results Obtained with Synthetic Data in Virtual Environments

A series of 3D virtual environments were designed to generate benchmark testing stereo sequences for human assistive devices. Since virtual scenes can provide ground truth information, testing using such synthetic data can offer valuable information about the

Fig. 4.5 Obstacles detection results for the KITTI dataset: overlaid segmentation results obtained in both $\theta$ and U-disparity images
4.1 Disparity based perception of 3D environments

Fig. 4.6 Virtual environment: Left image; Disparity map; Ground plane mask

Fig. 4.7 U-disparity and θ-disparity labeled for an image in the virtual environment sequence

efficiency of the algorithms and acknowledge worst case scenarios. Moreover, different environment scenarios can be tested without the need to physically find these locations or recreate some special situations in real-life environments.

The scenes were designed to mimic common outdoor locations. The development of the 3D scenes was done using the Unity game engine. Ground truth segmentation information was obtained by assigning a label and unique ID to each object in the scene. Thus, for the virtual environment testing scenarios the ground plane as well as camera orientation are straightforward to compute. Although accurate disparity information for each pixel is also available, the disparity maps were generated following the same procedure as for the real environments dataset.

Both U and θ-disparity segmentations were performed in the same manner (Figures 4.6, 4.7 and 4.1.1), using the same pre-processing parameters, except for the disparity threshold which was set to 10, as better disparity resolution was obtained in the virtual environment. The combined results of the θ and U-disparity can be observed in Fig. 4.9.

Fig. 4.8 Virtual environment: Polar representation of the disparity map after ground plane removal, θ-disparity before and after labelling
4.1.2 Ground geometry assessment

The key factor of distance computation is a correct evaluation of the disparities between the left and right image. Processing algorithms for the disparity map were initially reported in applications involving automotive and mobile robotics. In [129] an algorithm that analyses the V-disparity map is used to solve the obstacles detection problem for autonomous driving applications. In this work the goal is to identify free road space and obstacles. Wedel, et.al., [135], tackle the non-planarity of the road surface in real autonomous driving applications. They reveal an algorithm that uses flexible B-spline curves for piecewise planar or quadratic ground detection. The algorithm was evaluated mostly on highways were the free space in front of the vehicle is large and the B-spline curves can be accurately recovered.

The standard assessment for ground area is that it can be modeled as a plane. This assumption is true in applications that can generate dense disparity maps with low noise [136]. Otherwise, in conditions of non-uniform illuminations, sun glare, shadows, etc., the plane fitting involves multiple post filtering steps. These problems can be solved by modeling the ground using high order surfaces.

In the past two decades an important amount of work was reported on developing systems that can improve the quality of life for visually impaired. Visual impairment, or vision loss is a severe condition that seriously affects the life of the individuals suffering from it. Blind persons face challenges doing everyday things we take for granted, like reading or walking. One of the main aspects that underlie spatial navigation is the allocentric sense, the awareness of one’s body relative to the environment. For sighted people, the brain relies mostly on information from the eyes to accomplish the navigation tasks. One of the most important problems for blind people when moving in open or closed environments is precisely the lack of external references, creating a distortion of absolute directions, of the position of their heads and bodies related to free space and obstacles.
4.1 Disparity based perception of 3D environments

Regarding ground area identification for visually impaired applications most techniques assume a plane model. This plane model can be recover using a local approach, such as the one presented in [137], or a global one, as the one depicted in [138]. The local approach assumes the ground to be a plane that can be revered by exploring neighboring patches. The global one makes use of RANSAC for ground plane equation identification. Multiple variations of the RANSAC approach were reported later, each trying to improve the classic approach [139],[140].

In this paper we present a ground geometry assessment scheme using new disparity map analysis methods. Application wise, we evaluate different ground geometries which can further increase the robustness of obstacles and best free space detection. The targeted applications involve autonomous driving (KITTI database) and visually impaired assistive devices (Sound of Vision Project database). To the authors knowledge, this is the first attempt at evaluating outdoor ground plane geometry for visually impaired assistive solutions.

In this section, we detail some of the problems that emerge when dealing with ground recovery in complex stereo vision guidance applications. When the ground occupies a significant region in the image it usually corresponds to a dominant line in the V-disparity map and is associated with a planar surface. This is the reason for most of the reported works being on ground plane detection. An important role in establishing the accuracy of the ground plane detection is the quality of the disparity map, which is a key factor that will be further detailed in the experimental results subsection.

As a solution to these problems, a ground area detection scheme is proposed (fig. 4.1). The first step is to rectify the left and right images using the stereo system calibration parameters. The disparity map is computed using the rectified images. For the proposed scheme, the disparity map is recovered by employing the ELAS algorithm. The dense disparity map is further analyzed and the following ground identification algorithm is obtained.

An important implementation issue rises when the ground is not horizontal. This configuration does not have a direct correspondence in the V-disparity map, which makes the estimation of the ground using the Hough Transform a difficult task. High values of roll angle imply a counter rotation of the disparity map before generating the V disparity representation.

The roll angle of the camera can be recovered by using an extra sensor such as an Inertial Measurement Unit (IMU) [141] or using visual odometry (VO) [142]. Visual odometry is a technique for estimating the motion of a moving vehicle using video input from its on-board cameras. Once the camera motion is available, the values for the roll-pitch-yaw angles are recovered. Using the roll angle, the V-disparity image is rotated to be horizontally aligned, which ensures a robust ground area detection. Once the roll bias is corrected, the detection of
the ground area lines in the V-Disparity map can be done by employing the Hough Transform [132].

In order to identify the line in the V-disparity map that is the true correspondent to the ground area we first need to have a value for the tilt angle (pitch) of the stereo vision camera. For automotive applications the value of the pitch has a very small bias from the one obtained in the extrinsic calibration stage. For the visually impaired applications the estimation of the tilt angle tends to be more complex because of the higher number of degrees of freedom for the camera motion.

The initial value of the pitch can be obtained using [135]

\[
\tan \theta = \frac{h}{bf}d + \frac{1}{f}(c_y - v_y),
\]

(4.2)

where \(h\) is the ground related height of the stereo system, \(b\) is baseline, \(f\) is the focal length, \(c_y\) is the \(y\)-coordinate of the principal point in the image and the pair \((v_y, d)\) is a point in the V-disparity map. A robust estimate of the tilt angle using a set of V-disparity points is obtained by analyzing the histogram of the \(\tan \theta\) values calculated in (1). The tilt angle is found as the maximum in this histogram. In addition, as a quality measure, the variance and the number of V-disparity points supporting the found tilt angle are used.

For the resulted pitch, the perfect slope of the line from V-disparity that corresponds to the best ground area candidate is given by [143]:

\[
g = \frac{h}{b \cos \theta},
\]

(4.3)

Using the value of the perfect slope, the Hough transform can be used to recover all the straight lines that match a certain confidence interval in relation to \(g\).

To speed up the line detection algorithm and still maintain high robustness, one of the following two methods can be considered:

1. extract the line with the closest slope to \(g\) only from the lower half of the V-disparity map;

2. extract the line with the closest slope to \(g\) only from the lower quarter of the V-disparity map. Add points from V-disparity that verify the line equation;

The previous methods are employed under the assumption that the ground surface corresponds to a line that contains not have less than a user defined number of pixels located on each line.

Once the line is recovered from V-disparity the 3D fitting procedure can start. For the applications targeted by this work, we have chosen four types of surfaces, each with a
4.1 Disparity based perception of 3D environments

different number of parameters denoted by \((S \#\text{parameters})\):

\[
(S3) z = ax + by + c \\
(S4) z = ax + by + cxy + d \\
(S5) z = ax^2 + by^2 + cx + dy + e \\
(S8) z = ax + by + cxy + dx^2y + ey^2x + fx^2 + gy^2 + h
\]

Each of these surfaces will be evaluated in comparison with ground truth annotated stereo frames. The experimental results are presented in the next section.

The KITTI benchmark - Autonomous driving

The KITTI benchmark dataset [134] was designed for evaluating autonomous driving research algorithms. The autonomous platform was equipped with high resolution stereo PointGrey Flea2 camera systems. The design of the stereo system led to a baseline of roughly \(b = 0.54[m]\) between the cameras. The sequences were captured by driving on highways or around a mid-size city. Both static and dynamic objects appear in sequences. Manual labeled objects in 3D point clouds provide accurate 3D bounding boxes for multiple object classes including the ground area.

The SOV benchmark - Visually impaired people assistive devices

The goal of the Sound of Vision project is to design, implement and validate an original non-invasive hardware and software system to assist visually impaired people by creating and conveying an auditory representation of the surrounding environment. Regarding stereo vision acquisition, a stereo RGB camera - LI-OV580 from Leopard Imaging - is used for outdoor image capture. The two cameras are mounted on separate PCBs and are connected by wire to the central unit. The main advantage of this design is that the baseline can be configured specifically for the application. The Acquisition module captures Stereo frames, synchronizes them with the IMU data, rectifies the left and right images and then applies a stereo correspondence algorithm (Elas or SGBM) in order to compute the disparity map. Regarding resolution, the system acquires and rectifies stereo image pairs of a larger resolution, i.e., \(1280 \times 720\). In order to function properly, processing algorithms that make use of stereo images require the physical characteristics of the cameras. After calibration, the reported intrinsic parameters were: focal length \(f = 374.742\), projection point coordinates \(c_x = 403.958, c_y = 194.223\) and baseline \(b = 0.148221[m]\).
Experimental Results

In this section we reveal the evaluation results of the proposed ground surface recovery scheme. We target two types of applications: autonomous driving and visually impaired assistive awareness. Each of these applications is challenging, with a higher complexity degree for the assisting devices for visually impaired.
4.1 Disparity based perception of 3D environments

The standard assessment for ground area is that it can modeled as a plane. This assumption is true in applications that can generate dense disparity maps with low noise. Otherwise, in conditions of non-uniform illuminations, sun glare, shadows, etc., the plane fitting involves multiple post filtering steps (e.g. tuning of the height above ground threshold that decides whether a 3D point belongs to the ground or not).

The evaluation process consist in the following steps:

Fig. 4.11 First row: SOV - left and right frame, disparity map and ground truth labels
Second row: ground area for each surface from 1 to 4 obtained using method 1 3D points
Third row: ground area for each surface from 1 to 4 obtained using method 2 3D points
(i) Use the disparity map to compute the V-disparity representation

(ii) The two methods described for line recovery in section III are employed and two corresponding 3D point clouds are computed

(iii) For each of the two point clouds resulted in the previous step, fit the four surfaces (S3, S4, S5, S8) presented in section III.

(iv) Use ground truth labeled images of the tested frames to compute a fitting percentage.

This will reveal the accuracy of the line recovery methods and surface model fitting.

A Matlab implementation of the proposed ground surface fitting scheme was used for experimental results. In fig. 4.10 and fig. 4.11 two results of the proposed scheme are illustrated. In both cases we can observe that recovering the ground equivalent line from the V-disparity map using method 2 gives better results than method 1 for any of the considered surface models. Also, in comparison with classic planar fitting, adding $xy$ factors and squares of the individual $x$ and $y$ coordinates in the surface equation leads to more accurate results.

These conclusions are sustained by the numerical results that are depicted in Tables I, II and III. In these table we have the fitting percentages for three sequences: a sequence with 30 frames from KITTI, a second sequence with 25 frames from KITTI and a sequence with 35 frames from SOV.

<table>
<thead>
<tr>
<th></th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method 1</td>
<td>70.03%</td>
<td>70.22%</td>
<td>61.2%</td>
<td>75.13%</td>
</tr>
<tr>
<td>Method 2</td>
<td>71.3%</td>
<td>71.48%</td>
<td>62.24%</td>
<td>76.35%</td>
</tr>
</tbody>
</table>

Table 4.1 Kitti sequence 1, 30 frames

<table>
<thead>
<tr>
<th></th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method 1</td>
<td>81.55%</td>
<td>81.71%</td>
<td>74.69%</td>
<td>80.79%</td>
</tr>
<tr>
<td>Method 2</td>
<td>82.18%</td>
<td>82.15%</td>
<td>75.33%</td>
<td>82.45%</td>
</tr>
</tbody>
</table>

Table 4.2 Kitti sequence 2, 25 frames

For the KITTI sequences the highest percentage is obtained by the surface model S8, where we have the highest number of parameters. The percentage difference between method 1 and method 2 of v-disparity line detection is lower than 3%. This is a result of large baseline of the stereo acquisition system. In comparison, for the SOV sequence
4.2 Advanced visual based control techniques

4.2.1 Visual predictive control

The main goal of any image based control architecture is to drive a robot system using information acquired by a visual sensor [56]. If the considered robot system is a manipulator robot that has a video camera mounted on the gripper then a visual servoing structure with an eye-in-hand configuration is created. In order to achieve the main goal, a trajectory for the video camera must be designed. The trajectory is given by the integration of the camera velocity obtained from the IBVS architecture by minimizing the error between the current point features configuration and the desiring one. The error between point features in the image plane is defined as follows:

\[ e(k) = s(k) - s^*, \]  \hspace{1cm} (4.4)

where \( s(k) \) represents the point features position at \( k \)-iteration and \( s^* \) the desired configuration. Equation (4.4) is the general representation of the input for the image based controller. The output of the image based controller is the reference velocity \( v^*_{c} \) of the camera. If the task of extracting point features and finding the correspondences in the desired configuration is completed then a relationship between time variation of \( s \) and camera velocity can be generated. Let \( v^*_{c} \) be a vector composed from linear velocities of the camera frame and the instantaneous angular velocities of the camera frame. The link between the motion of point features and camera velocity \( v_{c} \) is given by [56]

\[ \dot{s} = L v_{c}, \]  \hspace{1cm} (4.5)
where by $L \in \mathbb{R}^{2n \times 6}$ denoted the image interaction matrix of $n$ point features that compose the vector $s$. From (4.4) and (4.5) it is obvious that:

$$\dot{e} = Lv_c.$$  \hspace{1cm} (4.6)

A classical proportional controller is defined [56] considering the static model of the velocities control loop, i.e $v_c = v_c^*$:

$$v_c^* = -\lambda \hat{L}^+ e,$$  \hspace{1cm} (4.7)

where $\hat{L}^+$ is the approximation of the pseudo-inverse of $L$.

In order to ensure better performances of a servoing system it is necessary to use advanced techniques of image based control. The model based predictive control approach for IBVS that is proposed in the present paper is developed for a 6 DOF robot manipulator with an eye-in-hand configuration for grasping a fixed object.

The presented scheme consists from a Virtual Cartesian Motion Device (VCMD), a Visual Servor (VS) and the Image based Predictive Controller (IbPC). Any predictive strategy is composed from the followings: a reference trajectory, a predictor of the future states of the process, a cost function and a minimization technique. In (Figure 4.12) it can be observed that as input the VCMD has the reference camera velocity denoted $v_c^*$ and as output the camera velocity screw $v_c$. The Visual Sensor has as input $v_c$ and generates the vector $s(k) = (u_i(k), v_i(k)), i = 1, n$ that stores the coordinates of the visual features from the image.

The main problem in designing a predictive control strategy is to have a good open loop model attached to the plant and also a suitable predictor for point features motion estimation in image plane having reference velocities of the video camera. This stage, the prediction model response, is fundamental in the predictive approach control of any automated system. The structure presented in (Figure 4.12) has an outer loop and an inner one. The inner one is the VCMD and it deals with robot joints control. In the following will be presented the plant model and also the point features motion prediction model.
4.2 Advanced visual based control techniques

Plant model

The plant for the considered visual servoing system is composed from a 6 d.o.f manipulator (representing the VCMD) and an eye in hand camera (VS). Next both the VCMD model and the VS model will be presented.

The VCMD model is nonlinear because the robot Jacobian \( J_r \) is a multivariable coupled one. The model of the robot dynamics is a function of the robot joints angular positions \( x_q \). One way to achieve a decoupled system is to employ a robust control strategy based on the joint space disturbance observer (DOB) and thus each joint axis is considered decoupled under the cut-off frequency of DOB [54]. The velocity controller is considered as a diagonal gain matrix:

\[
K_v = \text{diag}\{k_{v1}, k_{v2}, ..., k_{vn}\}.
\]  

(4.8)

If the non-singularity of the robot Jacobian \( J_r \) in the camera coordinates is assured, the transfer function from the acceleration command \( \dot{v}_c \) to the velocity \( v_c \) can be considered as an integrator system in the frequency region below the cut-off frequency.

The typical sampling period of the VCMD system is 0.2-1 [ms] and the typical cut-off frequency is 150-300 [rad/s]. Since the velocity controller is usually a proportional one with an amplification value \( k_v \), the inner loop system can be expressed in the frequency region below the cut-off frequency as:

\[
v_c(s) = G(s)v_c^*(s) = \frac{k_v}{s + k_v}I_6v_c^*(s).
\]  

(4.9)

Thus the following linearized discrete dynamic model for VCMD is obtained

\[
G(z^{-1}) = (1 - z^{-1})Z\left(\frac{G(s)}{s}\right),
\]  

(4.10)

where \( Z \) symbolizes \( z \) transform. The model (4.10) is an excellent approximation in the design of the IbPC controller since the cut-off frequency of the inner loop is much higher than the frequency of the outer loop.

Visual Sensor Model

Typically, the visual sensor is composed of a camera and an image processing block used to extract the features from the image. Considering the poses of the camera and object be \( x_c \in \mathbb{R}^6 \) and \( x_0 \in \mathbb{R}^6 \), the camera is modeled by the mapping

\[
f : \mathbb{R}^6 \times \mathbb{R}^6 \rightarrow \mathbb{R}^{2n},
\]  

(4.11)
where \( n \) point features are selected to characterize the object.

The camera is modeled using the mapping (4.11) as a function of \( x_c \) and \( x_0 \), the positions and the orientations of the camera and the object, respectively. In the sequel, it is considered that \( n \) point features were selected to characterize the object, being defined as

\[
s = [s_1^T, ..., s_n^T]^T,
\]

(4.12)

where \( s_i = [u_i, v_i]^T \) are the \( i \)-th feature coordinates in the image plane. The image processing block is modeled as

\[
s = g(f(x_c, x_o)),
\]

(4.13)

where \( g \) is a function that models the feature extracting algorithms.

In order to get the visual sensor model, the frames attached to the robot base \( F_b \), to the camera \( F_c \) and to the object \( F_o \), are considered as shown in (Figure 4.13):

![Figure 4.13 Camera, object and robot base frames](image)

Let \( T_c^b \) and \( T_o^b \) be the homogeneous transformations between the frames \( F_c \) and \( F_b \) and, respectively, \( F_o \) and \( F_b \). It is assumed that the object feature positions \( x_o^b \) related to the frame \( F_b \) is known and that the desired features \( s^* \) were extracted using a suitable operator. The homogeneous transformation \( T_c^b(0) \) for the camera start position is also considered known.

For getting the homogeneous transformation \( T_c^b(k) \), the camera velocity \( v_c(k) \) is integrated. The following homogeneous matrix is obtained

\[
T_c^b(k) = \begin{bmatrix} R_{RPM}(k) & t(k) \\ 0 & 1 \end{bmatrix},
\]

(4.14)
where: \( \mathbf{R}_{RPF} = \begin{bmatrix}
  c\phi c\omega & c\phi s\omega s\psi - s\phi c\psi & c\phi s\omega c\psi + s\phi s\psi \\
  s\phi c\omega & s\phi s\omega s\psi + c\phi c\psi & s\phi s\omega c\psi - c\phi s\psi \\
  -s\omega & c\omega s\psi & c\omega c\psi
\end{bmatrix} \), is the rotation matrix expressed using the roll, pitch and yaw approach, and \( \mathbf{t}(k) \) is the translation vector. The \( c(\cdot) \) and \( s(\cdot) \) notations represent the cosine and sine functions. Computing the inverse of \( T^b_c(k) \), the homogeneous transformation \( T^b_c(k) \) is derived and multiplying it with the object feature position \( \mathbf{x}_b^i \) related to the frame \( F_b \), the object feature position \( \mathbf{x}_j^c \) related to camera frame \( F_c \) is got. Having the feature position \( \mathbf{x}_j^c = [x_i, y_i, z_i]^T \), the intrinsic camera parameters \( p_x, p_y \) and the image center point coordinates \( (u_0, v_0) \), the point features coordinates expressed in pixels are obtained based on the perspective projection:

\[
\begin{align*}
  u_i &= \frac{x_i}{z_i} p_x + u_0; \\
  v_i &= \frac{y_i}{z_i} p_y + v_0.
\end{align*}
\]

Thus, the Visual Sensor model was found which has as input the camera velocity \( \mathbf{v}_c(k) \) and as output the point feature coordinates \( \mathbf{s}(k) \).

**Point features motion prediction model**

The most important stage in designing a predictive control strategy is the development of a suitable visual feature predictor. The main goal is to develop a valid model that can be used to predict the evolution of point features position in relation with the camera 3D motion. Let’s considered that any object from the image plain is described using point features. The definition of a point feature is typically taken to be a location in the image where the intensity function has a distinct peak. The image based predictor allows computing the future evolution of the image features over the prediction horizon \( h_p \). The prediction model must also be designed to satisfy the process constrains. Taking into account the relation between the derivative of point features and the camera velocity and the VCMD model from Subsection (4.2.1), for the first time in [57], a local model based predictor was developed by the authors (Figure 4.14). Later the same local model based predictor was presented in [58].

![Fig. 4.14 Plant model for prediction computing](image-url)
It is assumed that at every sampling period $T$, the visual sensor, composed of a camera and an image feature extraction block, gives the coordinates of the point features $(u_i(k), v_i(k))$, $i = 1, 4$ and it is possible to compute the depth $z_i(k)$ of the considered points with respect to camera frame. Using the discrete form of (4.5) and taking into account the linear discrete model of the VCMD (4.10) it results

$$s(k + 1|k) = s(k) + TL_k G(z^{-1})v^*_c(k)$$  \hspace{1cm} (4.16)

which represents the one-step ahead prediction of the image feature evolution. As known, for computing the $L_k$ interaction matrix at the discrete time $k$, information over the depth $z_i(k)$ is needed. For validation of the predictive model the information on depth can be approximated in two ways: at each iteration depth is considered to be $z^*$ the desired value or in case of knowing the 3D coordinates of the point features the depth is recovered after each sampling period from the current pose of the camera.

Shifting the one-step ahead prediction model (4.16) by recursion, the next predictors are obtained:

$$s(k + 2|k) = s(k + 1|k) + TL_{k+1} G(z^{-1})v^*_c(k + 1|k)$$

$$s(k + i|k) = s(k + i - 1|k) + TL_{k+i-1} G(z^{-1})v^*_c(k + i - 1|k)$$  \hspace{1cm} (4.17)

$$s(k + h_p|k) = s(k + h_p - 1|k) + TL_{k+h_p-1} G(z^{-1})v^*_c(k + h_p - 1|k)$$

The local model based predictor $s(k + i|k)$ is found based on the features predicted in the previous step $s(k + i - 1|k)$, also used to compute the image interaction matrix $L_{k+i-1}$, and on the future reference velocity $v^*_c(k + i - 1|k)$:

$$s(k + i|k) = s(k) + T \sum_{j=0}^{i-1} L_{k+j} G(z^{-1})v^*_c(k + j|k)$$  \hspace{1cm} (4.18)

The prediction is initiated with the features $s(k)$ at time $k$ that are obtained from the visual sensor VS using an appropriate feature detection operator.

**VPC architecture**

Having a suitable prediction model of the future behavior for point features the visual predictive control strategy can be developed. As illustrated in (Figure 4.12) three major components must be analyzed: the VCMD model, the VS model and the cost function that will be included in the Image based Predictive Controller (IbPC).
The aim is to solve, with better performances, the main problems of IBVS: minimization of the image error while taking into account the visual features visibility and robot constrains. For this goal the following cost function must be minimized:

$$J = \frac{1}{2} \sum_{i=1}^{h_p} e^T(k+i|k)Qe(k+i|k) + \delta \sum_{j=0}^{h_c} v^*_c(k+j|k)Wv^*_c(k+j|k) \quad (4.19)$$

The function in (4.19) is defined as a quadratic form of image control errors expressed in image plane measurements and camera velocity components. The parameters from (4.19) are the prediction horizon $h_p$, the command horizon $h_c$ and the weighting matrices $Q$ and $W$, which are positive defined.

The nonlinear relation between point features motion in image plane and camera motion in Cartesian space has a direct influence on the performances of the image based control law. Considering the classical approach or the predictive approach of any image based control law, the main issue is to ensure a realistic behavior of the camera motion so that the physically conditions of realizability can be fulfilled.

In order to obtain a desired dynamic behavior for the controlled output, in predictive control is introduced the reference trajectory. Such a reference trajectory for point features in image plane is depicted in (Figure 4.15) [54]. Beginning at the current discrete time $k$ with the current image $I(k)$ having the point features $s(k)$, a reference trajectory is designed from the image sequences $I(k+i)$, $i = 1, h_p$, with point features $w(k+i|k)$ so that $w(k+h_p|k) = s^*$ would be fulfilled. The notation $w(k+i|k)$ specifies that the reference trajectory depends on the initial conditions $s(k)$ at discrete time $k$. 

Fig. 4.15 Reference trajectory
Taking into account all the information described in the previous sections, a new type of visual predictive control architecture is proposed (Figure 4.16). The output of the reference trajectory generator must avoid image Jacobian singularities and local minima problems. Also the point features trajectories must be obtained from smooth camera velocity screws that fulfill the robot joints constraints. Different methods to generate camera velocity screw and transform it to point features trajectories were proposed, but neither were used as reference trajectory in visual predictive control.

Let’s consider an unknown fixed object described by a set of \( n \) coplanar (but not collinear) point features. If \( I(k) \) is the current image and \( I^* \) is the desired one, then the relation between the correspondent point features is given by \[ s_j^* = Gs_j(k), \quad j = 1, n, \] (4.20)

where \( s_j(k) = [u_j(k), v_j(k), 1]^T \) and \( s_j^* = [u_j^*, v_j^*, 1]^T \) and \( G \) is the collineation matrix.

The key in generating suitable point feature trajectories is the possibility of decomposing the \( G \) collineation matrix into a time dependent sequence of collineation matrices.

Let \( K = \begin{bmatrix} \delta_x & 0 & u_0 \\ 0 & \delta_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \) be the intrinsic camera parameters matrix. Assuming that a calibration stage was completed and \( K \) is known, the Euclidean homography

\[ H = K^{-1}GK \] (4.21)

can be recovered up to a scale factor. From (4.21) results that:

\[ G = KHK^{-1}. \] (4.22)
4.2 Advanced visual based control techniques

Let $F_k$ be the current camera frame and $F^*$ the desired one. The relative pose of $F^*$ in $F_k$ is described by a rotation matrix, $R_k^*$ and a translation vector $t_k^*$. The point features plane, denoted $\Pi$, has a normal vector $n_k$ and a distance $d_k$ related to the origin of $F_k$. Using the above notations the euclidean homography can be decomposed in:

$$H = R_k^*T_k(1 - \frac{t_k^*}{d_k}n_k^T),$$  \hspace{1cm} (4.23)$$

Because $K$ is constant, the decomposition of $G$ into time dependent sequence of collineation matrices can be solved by decomposing $H$ into a time depending sequence of matrices that fulfills: $H_\eta(0) = I$ and $H_\eta(1) = H$, [144]. Thus, if $q$ is a monotonic function of nondimensional time $\tau = \frac{t}{t^*}$, then the searched sequence of collineation matrices is:

$$G_\eta(q) = K H_\eta(q) K^{-1}$$  \hspace{1cm} (4.24)$$

Designing a smooth $q$ function can be done using different methods. In this paper a quintic-polynomial, as in [144], will be used:

$$q(\tau) = a_5\tau^5 + a_4\tau^4 + a_3\tau^3 + a_2\tau^2 + a_1\tau + a_0$$  \hspace{1cm} (4.25)$$

The difference from the design of $q$ proposed in [144], is the linear system used to derive the $a_i$ coefficients:

$$\begin{cases} 
q(0) = 0; & \dot{q}(0) = \alpha; & \ddot{q}(0) = 0 \\
q(t^*) = 1; & \dot{q}(t^*) = 0; & \ddot{q}(t^*) = 0 
\end{cases}$$  \hspace{1cm} (4.26)$$

Once $q(\cdot)$ was computed, the time dependent Euclidean homography $H_\eta(q)$ is given by:

$$H_\eta(q) = R_\eta(q)^T(I - \frac{t_\eta(q)}{d_i}n_i^T).$$  \hspace{1cm} (4.27)$$

As in [144], the rotation matrix $R_\eta(q)$ can be expressed through a time depending axis-angle representation of the matrix $R_k^*$. The time dependent reference scaled translation of the camera, $\frac{t_\eta(q)}{d_k}$, is a trajectory that identifies the position of the reference camera frame origin during the task. In [144] a helicoidal shape is proposed and the reference scaled position during the translation is extracted from a generic helix analytical form. Starting from the current configuration, the reference trajectory of each point feature is generated:

$$w(k+i|k) = G_\eta(q)s(k), \hspace{1cm} i = 1,h_p,$$  \hspace{1cm} (4.28)$$

so that at the end of the prediction horizon to obtain $w(k+h_p|k) = s^*$. 
Having the reference trajectory \( w \) defined, the cost function (4.19) is changed into:

\[
J = \frac{1}{2} \sum_{i=1}^{h_p} (s(k+i|k) - w(k+i|k))^T Q (s(k+i|k) - w(k+i|k)) + \delta \sum_{j=0}^{h_c} v^T (k+j) W v (k+j).
\]

(4.29)

A closer inspection of (4.29) reveals that \( h_p, h_c, \delta \) are not the only tuning parameters of the cost function, but also \( \alpha \). This new tuning parameter plays a very important role because its values generates different behavior of the time function \( q(\cdot) \) and thus the desired behavior of the controlled variable can be obtained.

### 4.2.2 Nonlinear visual servoing systems using image moments

Image moments represent generic visual features that can be computed easily from a binary or a segmented image, or from a set of extracted points of interest [60]. Computing image moments in a frame sequence is a more stable and robust process than detecting point features, due to the fact that only a high level global matching stage of the object is needed and not an accurate matching of each object point. This type of descriptors can be used to characterize different geometrical entities whatever the object shape complexity. One of the problems when using point features or lines in visual servoing applications is the high coupled state of the interaction matrix. By including image moments in visual feedback control a decoupling of the interaction matrix was obtained for the case of objects parallel to projection plane [62, 145].

In order to improve the behavior of the image based control laws, advanced approaches from control theory were adapted to visual servoing systems [57, 58, 54]. From control theory it is known that any predictive strategy is composed from the followings: a model based predictor of future behavior of the process when different control scenarios are considered, a reference trajectory, a cost function and a minimization technique. Two types of model based predictors for the process were considered: global and local. The local model is based on the interaction matrix while the global one is based on the knowledge of the 3-D target coordinates in the camera frame.
Image Moments Interaction Matrix

The analytical form of the interaction matrix for the general case is given in ([62]). For the discrete case the time derivative of $\mu_{ij}$ is obtained by differentiation:

$$
\dot{\mu}_{ij} = \sum_{k=1}^{n} i(x_k - x_g)^{i-1}(y_k - y_g)^i(\dot{x}_k - \dot{x}_g) + j(x_k - x_g)^j(y_k - y_g)^{j-1}(\dot{y}_k - \dot{y}_g). \tag{4.30}
$$

Using (4.30) the interaction matrix of the centered moments $\mu_{ij}$ is computed by:

$$
L_{\mu_{ij}} = [\mu_{v_x}; \mu_{v_y}; \mu_{v_z}; \mu_{\omega_x}; \mu_{\omega_y}; \mu_{\omega_z}]. \tag{4.31}
$$

The detailed structure of the parameters from (4.31) can be found in ([145]). For the set of image moments $f_m = [x_n, y_n, a_n, \tau, \xi, \alpha]^T$ the interaction matrix is:

$$
L f_m = \begin{bmatrix}
-1 & 0 & 0 & a_n e_{11} & -a_n (1 + e_{12}) & y_n \\
0 & -1 & 0 & a_n (1 + e_{21}) & -a_n e_{11} & -x_n \\
0 & 0 & -1 & -e_{31} & e_{32} & 0 \\
0 & 0 & 0 & \tau_{\omega_x} & \tau_{\omega_y} & 0 \\
0 & 0 & 0 & \xi_{\omega_x} & \xi_{\omega_y} & 0 \\
0 & 0 & 0 & \alpha_{\omega_x} & \alpha_{\omega_y} & -1
\end{bmatrix}. \tag{4.32}
$$

The analytical form of the parameters from 4.32 can be found in [145].

Virtual Cartesian Motion Device

As stated, Fig. 4.17 is an image moments based predictive control architecture that can be used to control a 6 d.o.f. robot manipulator with an eye-in-hand configuration. The manipulator can be modeled as a decoupled Virtual Cartesian Motion Device (VCMD). One way to achieve a decoupled system is to employ a robust control strategy based on the joint space disturbance observer (DOB) ([146]) and thus, each joint axis is considered decoupled under the cut-off frequency of DOB.

Considering the robot manipulator to be a decoupled linear diagonal plant, then, the camera velocity can be computed in the camera coordinates using:

$$
v_c(s) = G(s)v'_c(s). \tag{4.33}
$$
The input of the VCMD is the camera velocity reference \( v^*_c \) which is applied to analog velocity control loops described by the transfer matrix \( G(s) \) through a zero-order holder (ZOH). The output of the VCMD has a sampling period \( T \) thus resulting \( v_c(k) \). Assuming that the velocity controller is designed as a diagonal matrix \( k_v = \text{diag}\{k_v, \ldots, k_v\} \), the following linear discrete model for VCMD is obtained

\[
G(z^{-1}) = (1 - z^{-1}) \left( \frac{G(s)}{s} \right); \quad G(s) = \frac{k_v}{s + k_v} I_6, \tag{4.34}
\]

where \( k_v \) is a proportional gain controller and represents the \( z \) transform. Taking into account the decoupling effect of the joint level velocity loops, \( G(z^{-1}) \) can be rewritten as

\[
G(z^{-1}) = \text{diag}\left\{ \frac{b_i(z^{-1})}{a_i(z^{-1})} \right\} = A^{-1}(z^{-1})B(z^{-1}), i = 1, 6, \tag{4.35}
\]

where the polynomials matrices \( A \) and \( B \) are equal to:

\[
A(z^{-1}) = \text{diag}\{a_i(z^{-1})\} \\
B(z^{-1}) = \text{diag}\{b_i(z^{-1})\}. \tag{4.36}
\]

Based on (4.34) a robust image moments based predictor was developed. Its’ properties are analyzed next.

**Image Moments based Predictor**

In order to have a complete description of the visual based control law in the image moments space, an image moments based predictor is needed. Based on the assumption that the interaction matrix related to image moments has slow changes around the desired camera Reference trajectory generator, the VCMD optimization block, the local model based predictor, and the visual sensor are connected as shown in Fig. 4.17 Visual predictive control architecture.
pose [145], a new local model based predictor was designed using directly the image moments features (Fig. 4.18).

The discrete time relation between camera velocity and the image moments velocity is obtained from

$$\dot{f}_m = L_{f_m} v_c,$$

(4.37)

by discretization using Euler’s method:

$$f_m(k+1) = f_m(k) + T L_{f_m}(k)v_c(k).$$

(4.38)

The variable $L_{f_m}(k)$ is the interaction matrix related to a set of image moments $f_m$ derived from point features acquired at the current discrete time $k$ with a camera and an appropriate point feature detector.

A robustness issue emerges when dealing with small neighborhoods around the desired configuration. In those neighborhoods there is the possibility of reaching a so called singularity, where the generated camera velocity is 0 but the desired configuration is not reached. Let $r = (r_x, r_y, r_z) = \theta u$ be the $(3 \times 1)$ vector containing the axis of rotation $u$ and the angle of rotation $0 \leq \theta < 2\pi$. Combining the translation vector $t$ with vector $r$ results $\gamma = (t, r)$, a $(6 \times 1)$ vector which contains the pose of a reference frame expressed in an open subset $S \subset \mathbb{R} \times SO(3)$, where $SO(3)$ [147] is the special orthogonal group of $3 \times 3$ matrices.

Let $\gamma_c \in S$ and $\gamma_o \in S$ be the camera pose and the object pose. Based on the camera pose dependence of the visual features, the equation (4.37) can be written as:

$$f_m(\gamma_c) = \frac{\partial f_m}{\partial \gamma_c}$$

(4.39)

Fig. 4.18 Image moments predictor
From the assumption $f_m(\gamma_c) \neq f_m^*$ results that the Taylor expansion of $f_m(\gamma_c)$ around a neighborhood of $\gamma_c^*$ is:

$$f_m(\gamma_c) = f_m(\gamma_c^*) + \frac{\partial f_m}{\partial \gamma_c}(\gamma_c - \gamma_c^*). \quad (4.40)$$

The time differentiation of (4.40) gives

$$\dot{f}_m = L_{f_m} \mathbf{v}_c, \quad (4.41)$$

which combined with (4.37) leads to:

$$\dot{f}_m = \frac{1}{2} (L_{f_m} + L_{f_m}^* \mathbf{v}_c. \quad (4.42)$$

Taking into account performance analysis conducted until now, the interaction matrix $L_{f_m}$ used by the image moments predictive controller will be replaced with \( L_{f_m} = 1/2 \left( L_{f_m} + L_{f_m}(k) \right) \), where $L_{f_m}$ is the interaction matrix related to $f_m^*$. The one-step ahead prediction of the image moments evolution, when the plant model is considered, can be computed using (4.38) and the discrete model (4.35) of the VCMD, resulting:

$$f_m(k+1|k) = f_m(k) + T L_{f_m}(k) A^{-1}(z^{-1}) B(z^{-1}) \mathbf{v}_c^*(k|k). \quad (4.43)$$

Shifting the one-step ahead prediction model (4.43) by recursion and having the hypothesis that the interaction matrix $L_{f_m}$ is constant over the prediction horizon $h_p$, the next predictors over prediction horizon $h_p$ are obtained:

$$f_m(k+i|k) = f_m(k) + T L_{f_m}(k) A^{-1}(z^{-1}) B(z^{-1}) \sum_{i=1}^{h_p} \mathbf{v}_c^*(k+i-1|k). \quad (4.44)$$

The prediction is initiated with the image moments $f_m(k)$ derived from point features or edges extracted at discrete time $k$.

This predictor is used to construct a reference trajectory based cost function, which is the input in the optimization block (Fig. 4.17).

**Reference Trajectory based Cost Function**

The aim of the optimization block from Fig. 4.17 is to generate system inputs so that the controlled outputs converge to a desired set point. For that, we define a cost function $J$ as a
4.2 Advanced visual based control techniques

A quadratic function of predicted control errors and inputs. The cost function is

\[
J = \frac{1}{2} \sum_{i=1}^{h_p} e^T (k+i|k) Q e(k+i|k) + \sum_{i=0}^{h_u-1} v^*_e (k+i|k) W v_e (k+i|k),
\]

(4.45)

where \( Q \) and \( W \) denote positive definite, symmetric weighting matrices, \( h_p \) is the prediction horizon and \( h_u \) is the control horizon.

The error is defined by

\[
e(k+i|k) = f_m(k+i|k) - w_m(k+i|k),
\]

(4.46)

where \( w_m(k+i|k) \) is an image moments reference trajectory related to the current step \( k \).

A reference trajectory design directly in the image moments space is presented next. Let \( q = q(\sigma) \) be a monotonic function of the non-dimensional time \( \sigma = i/h_p \), which starts from \( q(0) \) for \( i = kT \) and ends at \( q(1) \) for \( i = (k+h_p)T \). This function can be approximated by a quintic-polynomial

\[
q(\sigma) = a_5 \sigma^5 + a_4 \sigma^4 + a_3 \sigma^3 + a_2 \sigma^2 + a_1 \sigma + a_0,
\]

(4.47)

which emerges from the initial and final conditions:

\[
\begin{align*}
q(0) &= 0; & \dot{q}(0) &= \psi; & \ddot{q}(0) &= 0 \\
q(1) &= 1; & \dot{q}(1) &= 0; & \ddot{q}(1) &= 0
\end{align*}
\]

(4.48)

Tuning the \( \psi \) parameter will generate different behaviors of the time function \( q \). The geometrical interpretation of \( \psi \) is the tangent of the angle under which the time function \( q \) starts. Thus if the value of \( \psi \) is increased then a faster response for \( q \) is obtained, this implies that \( \psi \) is a parameter that will influence the dynamic of the resulting path which is defined by

\[
w_m(k+i|k) = f_m(k) + q \left( \frac{i}{h_p} \right) \Delta, \quad i = 1, h_p,
\]

(4.49)

where \( \Delta = f^* - f_m(k) \).

In order to completely define the cost function \( J \) constraints must be added to (4.45). The main constraints are associated to the limits of the image. These are called visibility constraints and are used to ensure that all the features remain visible throughout the entire servoing task:

\[
(x_j(k), y_j(k)) \in [(x_{min}, y_{min}), (x_{max}, y_{max})], \quad j = 1, n.
\]

(4.50)
Also, a visual servoing system may be subject to torque constraints or joints boundaries.

### 4.2.3 VPC architecture for industrial robots

In order to analyze the performance of the VPC strategy described in Section 3 a real-time visual servoing architecture for a FANUC 6 d.o.f robot manipulator was developed. The proposed servoing architecture (Fig. 4.19) is composed from three different modules: Image Based Control Strategy, Robot Communication Interface (RCI) and Robot Controller, each one having a different execution time. The following notations are considered: \( T_{ap} \) the execution time for the Image Based Control module, \( T_c \) the RCI module execution time and \( T \) the execution time for Robot Controller module. The sample period \( (T_s) \) of the entire architecture is obtained from:

\[
T_s = T_{ap} + T_c + T.
\]  

(4.51)

This architecture is an event driven one, the acquisition of an image being the event that triggers the controller to generate new information.

The aim of direct kinematics is to compute the pose of the camera frame, attached to the last link, as a function of the joint variables. In this paper the considered robot is an open chain that has 6 revolute joints and thus the direct kinematic equation will be:

\[
{^0T_6}(q) = {^0A_1}(q_1){^1A_2}(q_2)...{^5A_6}(q_6),
\]  

(4.52)

where \( ^iA_i(q_i) \), \( i = 1, 6 \) are homogeneous transformation matrices, each of which is a function of a single joint variable. In order to compute (4.52) the Denavit-Hartenberg (D-H) convention ([147]) was used.

The inverse kinematics problem consists of the determination of the joint variables corresponding to a given camera frame pose ([147]). The existence of solutions is guaranteed.
only if the given camera frame pose belong to the manipulator dexterous workspace. An
unique solution to the inverse kinematic problem is almost impossible to obtain. One can use
either analytical approaches or numerical methods ([147]).

In order to have a direct communication between the Image Based Control Strategy
module, which is implemented in Matlab, and the R-J3iB controller a Robot Communication
Interface (RCI) is used. The RCI module has three blocks: I/O, User Interface and TCP/IP.
The RCI incorporates the User Interface block which ensures the communication with the
robot controller. This one allows external interaction through the TCP/IP via Ethernet network
and the effective communication is carried out using Robot Server from PC Developer’s
Kit. The role of RCI in the visual servoing architecture is to analyze the MATLAB output,
transfer the proper data to the robots controller and read the new configuration of the joints.

Robot Controller is a module composed from all the entities that ensure the correct
motion of the robot in order to reach a desired configuration of the joints. The data flow of the
proposed architecture is the following: first the image based predictive controller generates
the camera reference velocity $v^*_c$ using the method described in Sections 2 and 3. Next an
integral action is applied and the desired pose $x^*_c(t)$ is obtained and further transformed into
a homogeneous matrix $\Delta^T c^*$ that represents the difference between the current pose $c^T b(t)$
of the camera frame and the desired one. The product of $\Delta^T c^*$ with $c^T b(t)$ generates $c^T b^*$ that is
the input of the IKM. The solution of IKM, denoted by $q^*$, is the new current configuration
of the joints. This information is written in a data file which is read by the User Interface and
sent to the robot controller.

The visual servoing control architecture presented in (Fig. 4.19) was implemented and
tested on a real-time visual servoing system (Fig. 4.20) composed from a master PC, a Fanuc
robot with a R-J3iB controller, a video camera and a work table. The experimental results
validate the visual servoing control architecture. Based on an Ethernet network, the master
PC is directly connected with the robot controller. For the Fanuc robot the D-H parameters
are presented in Table 1.

<table>
<thead>
<tr>
<th>Joint</th>
<th>$\alpha$</th>
<th>$a$</th>
<th>$\theta$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$-\pi/2$</td>
<td>0.15</td>
<td>0</td>
<td>0.45</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0.36</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>$-\pi/2$</td>
<td>0.15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>$-\pi/2$</td>
<td>0</td>
<td>0</td>
<td>0.35</td>
</tr>
<tr>
<td>5</td>
<td>$\pi/2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0.07</td>
<td>0</td>
<td>0.11</td>
</tr>
</tbody>
</table>
Using the visual control architecture designed in Section 3, different experiments were conducted for proportional and predictive with reference trajectory image moments based controllers. The visual proportional control law based on image moments is given by ([56]):

$$v_c(k) = -\lambda \hat{L}_{\text{tm}}^+ (f_m - f_m(k)^*)$$

(4.53)

where $\hat{L}_{\text{tm}}^+$ is the pseudo-inverse of $\hat{L}_{\text{tm}}$ and the gain factor $\lambda$ was tuned to obtain the best performances. The weighting matrices $W$ and $Q$ play an essential role in the predictive controllers’ performances, and the chosen structures ensure a better tracking of the reference trajectory. The parameters of the visual predictive controller were tuned as follows: $h_p = 3, h_u = 1, W = I_6$ and $Q = e^{1-i}I_6, i = 1, h_p$. This structure of the $Q$ matrix ensures weighting the error at each sampling instant more and more over the prediction horizon, thus stressing the error at the first step in the predictive sequence. For the image moments predictive controller the minimization of the cost function was done using the $fmincon$ function from Matlab. The $fmincon$ parameters were set to keep the camera velocity between $[-0.5; 0.5] \frac{m}{s}$ and $[-11.25; 11.25] \frac{deg}{s}$. 
4.2 Advanced visual based control techniques

Fig. 4.21 Point features configuration for general object: (a) Initial; (b) Desired

In the first experiment an object which can be described by a small set of point features was chosen. The point features were extracted via Harris algorithm presented in Subsection 2.1. Images were acquired using a Foculus FO124TC video camera. This type of industrial camera is based on a serial communication interface IEEE-1394a (FireWire), which allows a transfer rate of 400 Mbit/s and uses a 6-pin connector. For the acquisition process the image was chosen to be monochrome, while the resolution was set on $640 \times 480$. A calibration stage was conducted using the Calibration Toolbox for Matlab and the intrinsic parameters matrix was revealed to be equal to:

$$
K = \begin{bmatrix}
838.85 & 0 & 302.7 \\
0 & 837.79 & 272.49 \\
0 & 0 & 1
\end{bmatrix}
$$

(4.54)

The desired depth is $Z^* = 0.3$ m while the time parameters were set as follows: $T = 0.1s$, $T_c = 0.25s$ and $T_{ap} = 2.12s$. In Fig. 4.21(a) is presented the initial configuration while the desired one is depicted in Fig. 4.21(b). The linear and angular camera velocities obtained for the proportional approach are presented in Fig. 4.22(a), while for the predictive method the results are depicted in Fig. 4.22(b). The experimental results showed that all the considered constraints were fulfilled by both controllers and a 10% improvement in the number of iterations needed to reach the zero steady state was achieved by the predictive controller.

For the second experiment a symmetric object is considered. The object is described by its entire region and so the image moments computation becomes a highly time consuming stage. In order to improve this aspect, a Sony XCD-V60CR visual sensor was used. The link between this type of industrial camera and the master PC is realized by an IEEE-1394b serial bus interface which allows a transfer rate of 800 Mbit/s and uses a 9-pin connector.
An initial image Fig. 4.23(a) and a desired one Fig. 4.23(b), are the inputs of the experiment. Both desired and starting camera pose are parallel with the object plane, the
4.2 Advanced visual based control techniques

Fig. 4.24 The linear and angular camera velocity for: a),c) image moments based proportional control law; b),d) image moments based predictive control law with reference trajectory.

desired depth is $Z^* = 0.19 \, m$ and based on the time parameters $T = 0.12\, s$, $T_c = 0.24\, s$ , $T_{ap} = 1.27\, s$ the sample period was set on $T_s = 1.65\, s$. The time evolution of the camera velocities is presented in (Fig. 8). The proportional image moments based controller offers the results from Fig. 4.24(a) and Fig. 4.24(b), while in Fig. 4.24(c) and Fig. 4.24(d) are the results of the image moments based predictive controller.

Both experiments were successful and in each case convergence was achieved. The linear velocities generated by the proportional controller are comparable with the ones generated by the predictive controller. The number of iterations needed by the predictive controller to reach the desired configuration is 15% lower then the proportional approach. Another interesting aspect emerges when the angular velocities of the proportional controller are compared with the ones of the predictive controller. During the experiments the angular velocities for proportional controller presented some oscillations which can generate execution problems to the robot manipulator.

Thus taking into account the lower number of iterations needed to complete the servoing task and the smoothness of the generated camera velocities, we conclude that the best performances were obtained using the image moments based visual predictive controller with reference trajectory.
Chapter 5

Overview of professional and scientific results

5.1 Education and training

Educational background


- 2005–2008: Bachelor Scholar in Mathematics at Alexandru I. Cuza University, Faculty of Mathematics, Romania.


International stages

*February – June 2012*, 4 moths collaborative visit to Intelligent Systems and Networks Group, Imperial College London, United Kingdom, coordinator Professor Yiannis Demiris. Research theme: Potential applications for the iCub platform using visual servoing techniques.

*February – March 2011*, 2 moths collaborative visit to Robotics Intelligence Lab, Jaume I University, Castello de la Plana, Spain, coordinator Dr. Enric Cervera. Research theme: Real-Time Visual Predictive Control of Manipulation Systems.
Overview of professional and scientific results


October 2007, 1 month collaborative visit to Robotics Intelligence Lab, Jaume I University, Castello de la Plana, Spain, coordinator Dr. Enric Cervera. Research theme: Development and implementation of visual servoing algorithms.

5.2 Professional

Oct. 2014 – Present Associate Professor, Dept. of Automatic Control and Applied Informatics, Faculty of Automatic Control and Computer Engineering, Gheorghe Asachi Technical University of Iasi (Romania)
Oct. 2012 – Sept. 2014 Assistant Professor, Dept. of Automatic Control and Applied Informatics, Faculty of Automatic Control and Computer Engineering, Gheorghe Asachi Technical University of Iasi (Romania)
June 2010 – March 2013 Postdoctoral Researcher, Faculty of Automatic Control and Computer Engineering, Gheorghe Asachi Technical University of Iasi (Romania), Research theme: Advanced control techniques for visual servoing systems
Oct. 2008 – May 2010 Teaching Assistant, Dept. of Automatic Control and Applied Informatics, Faculty of Automatic Control and Computer Engineering, Gheorghe Asachi Technical University of Iasi (Romania)

5.3 Teaching

Current teaching activities include:
- Undergraduate courses: System Identification, Statistics and Data Processing
- Undergraduate laboratory assignments: Artificial Vision Systems
- Graduate courses: Modeling and Design of Robot Systems, Visual Servoing Control Systems

5.4 Domains of competence and research activity

Fields of interest visual servoing control systems, computer vision, rigid body motion parameterization, robotics, intelligent systems.
5.5 Publications

Research grants

- Member of TRADE-IT - PCCDI national research grant, title: *Innovative technologies for advanced materials recovery from IT and telecommunication waste* (2018-2020)

- Member of MRCA - PNIII TE national research grant, title: *Cooperative navigation of mobile robots in complex applications* (2018-2020)

- Coordinator of a young research team grant (CNCSIS-TE2016 competition) financed by the Technical University of Iasi

- Member in Sound of Vision - a H2020 international research grant, title: *Natural sense of vision through acoustics and haptics* (2015-2018)

- Member of a POS-CCE research grant, title: *Improvement of the diagnostic-therapeutic medical process through the use of an integrated system of clinical-paraclinical data management* (2014-2015)

- Member in a Brancusi Grant, Joint research project Romania – France, title: *Commande prédictive coopérative des systèmes complexes. Modélisation et gestion d’énergie pour le bâtiment intelligent*, (2009-2010)

- Member of research grant awarded by Romanian Research Council (CNCSIS Type A project), title: *Artificial vision techniques for manipulator robot control*, (2007-2008)

- Coordinator of young researcher grant awarded by the Romanian Research Council (CNCSIS-TD: 2007-2008) title: *Image processing algorithms for object motion tracking*

<table>
<thead>
<tr>
<th>Publications type</th>
<th>#</th>
<th>Clarivate WoS</th>
<th>Q1/Q2</th>
<th>Citations</th>
<th>H-index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Journal articles</td>
<td>20</td>
<td>14</td>
<td>5/3</td>
<td>WoS</td>
<td>167</td>
</tr>
<tr>
<td>Conf. articles</td>
<td>52</td>
<td>27</td>
<td>-</td>
<td>Scopus</td>
<td>325</td>
</tr>
<tr>
<td>Books / Chapters</td>
<td>6</td>
<td>3</td>
<td>-</td>
<td>Scholar</td>
<td>463</td>
</tr>
</tbody>
</table>

Publons - Adrian Burlacu
Scopus - Adrian Burlacu
GScholar - Adrian Burlacu
List of 10 important publications


- Condurache D., Burlacu A., Onboard Exact Solution to the Full-Body Relative Orbital Motion Problem, Journal of Guidance Control and Dynamics, Vol. 31(12),2638-2648, 2016 (F.I. 2.024)


- Burlacu A., Lazar C., Reference Trajectory Based Visual Predictive Control, Advanced Robotics, Vol.26, No.8-9, 2012 (F.I. 0.961)

5.6  Additional information

Awards

PhD Cum Laude "Image processing algorithms for object motion tracking", PhD thesis in Computer Science, 2009

Research scholarship (2006-2008) offered by Siemens PSE Brasov, Romania


PRECISI UEFISCDI - 8 awards for Q1 / Q2 journal articles

Other activities

Co-promoter of a PhD Thesis - Cosmin Copot, Control Techniques for Visual Servoing Systems, coordinator prof. Corneliu Lazar. The defense of the thesis was held on Nov. 18th 2011 at Faculty of Automatic Control and Computer Engineering, Gheorghe Asachi Technical University of Iasi, Romania

Member of advisory PhD commission for both national and international students (ERASMUS Mundus Joint Doctorates).

Regular reviewer for journals such as IEEE Transactions on Industrial Electronics, IEEE Transactions on Automation Science and Engineering, Robotics and Autonomous Systems, Journal of Intelligent & Robotic Systems, IET Control Theory & Applications;

Scientific reviewer of high impact conferences such as IEEE International Conference on Robotics and Automation (ICRA), IFAC world congress, International Conference on Computer Vision (ICCV)


Organizer of the invited session “Human Computer Interaction” – ICSTCC 2017, co-chairs: Runar Unnthorsson, Pawel Strumillo, Simona Caraiman
Member of organizing, program or technical committees for international conferences: Int. Conf. on Autonomous Robot Systems and Competitions (ICARSC 2016-2019); Int. Conf. on System Theory, Control and Computing (ICSTCC 2014-2019); Int. Conf. on Automation, Quality and Testing, Robotics (AQTR 2014, 2016, 2018)
Chapter 6

Future career development

Throughout all my activities at the Faculty of Automatic Control and Computer Engineering, Gheorghe Asachi Technical University of Iasi, my teaching career has been supported by the individual and contractual research. Thus, a permanent development of the curricula and practical activities has been achieved by using concepts and new elements resulting from the research. Consequently, I believe that the academic career development can be obtained based on the previous results and on the expertise gained in both areas of activity: teaching and research.

Teaching Career

The evolution of the teaching career is developing within certain boundaries of the institutional guidelines. Nevertheless, different interactions with similar international institutions, laboratories, personnel and also industry and labour market will continuously modify the perception of the teacher as well as the curricula. The control systems field has its own dynamic superior to other scientific and academic branches, generating a permanent need of updating with the achievements of the industry. The expertise accumulated up to this moment together with the industry and students feedback enable to generate an overall view of my future teaching activities.

Regarding teaching activities, one of the main goals is to propose new directions in both the undergraduate and graduate programs based on my expertise as well as in accordance with the faculty/university specialities and also with the ARACIS recommendations. Another goal will be permanent concern for improvement of teaching activities in the sense of knowledge transfer improvement such that to efficiently combines theoretical background with new methods and techniques as well as with their related applications within my expertise field. Last but not least, continuation of the teaching activities as part of the cooperation on bilateral
academic agreements within the framework of the European Union mobility actions for academic staff.

The self-developed research activities of the undergraduate and graduate students are essential to their professional formation as independent decision makers and solutions providers. Therefore, I integrated the experimental activities in the License projects and Dissertation thesis as a main component of the work. That enabled the observation of the young researchers’ potential and also built up a selection base for the PhD students recruiting.

Research Career

Future research activity will aim to develop original elements with a high potential, in order to obtain several impact results in the field of robotics and close related areas. In the near timeline, the robots are expected to be a very reliable asset in dynamic environments such as industry, health or domestic. The variability in these environments and the close interaction and collaboration with humans generates the necessity of developing new techniques to increase perception and control and new methods for programming and reprogramming robotic platforms in an intuitive and fast way.

As a general theme for the upcoming three to five years, I will start from the research presented in this thesis. The following major topics will be addressed:

- **Human to robot skill transfer**

  A skill is defined by a specific set of motions done by the human. To understand human motions and to successfully generalize motions according to the changing state of the environment, an internal representation of the motion is required in the form of a motion model. Invariant representations deliver compact and flexible motion primitives from which trajectories can be generated in a wide variety of settings or on different robot platforms. An invariant motion trajectory representation contains only the coordinate-free information eliminating the dependence on unwanted motion variations such as magnitude, execution time or velocity along the trajectory. All these variations make it difficult to recognize motions and to generalize from demonstrated motions.

  The innovative character of the research will come from the original motion representation and recognition techniques based on dual Lie algebras. This approach is different from the current state of the art methods used for motion parameterization in applications for learning by demonstration. The main difference is the capacity of the new techniques to consider both translation and rotation in a complete algebraic
framework that allows for coordinates-free solutions to motion recovery problems. This advantage is one of the strong points of the research, taking into account that the state of the art consists mainly of solutions developed in SE3 which forces the algorithm to decouple rotation from translation.

• **Position based visual servoing - a new approach**

This approach will be different from existing methods due to the lack of need for a pose recovery based on a prior known 3D model of the object. Having a visual sensor with depth capabilities, we consider as features a set of points identified in both the current and desired view. The 3D positions of these feature points will be recovered and passed as input in the future solution.

The use of point features is similar to the IBVS technique. However, the new solution does not need a nonlinear operator, such as the interaction matrix, to link the visual sensor motion with the feature points motion. Thus the number of feature points can be as high as wanted without the disadvantage of increasing the size of an operator while expanding its kernel. The core of the solution will be built from the possibility of imposing a desired velocity field over the set of feature points. This velocity field is further characterized by a dual screw vector which embeds the angular and linear velocity of the visual sensor. Similar to the existing visual servoing techniques, the imposed velocity field can be tuned using a gain parameter.

Validation, laboratory demonstration and practical use will be carried in both robotics and intelligent systems applications.

• **Applications for robotic systems**

One of my goals is to apply the results of my research in practice. I have already been involved in several projects which targeted real-time applications and many of the topics are strongly motivated by practical application. I will validate the fundamental and algorithmic contributions in real-life case studies. Cooperation with industrial partners will be sought, with the longer-term goal of commercial deployment.

**Doctoral School Activities**

Correlated with the didactic and research activities, one of my future goals is to start the activity as PhD advisor in the Doctoral School. I will try to attract and to supervise high quality PhD students in the aforementioned research areas. As I already stated in the overview chapter, I have been involved in the evolution of some PhD students of my PhD advisors
colleagues. Therefore, I intend to continue this and to strengthen the PhD activities. This work will be sustained by the coordinated research grants and projects, which will provide the necessary financial support for PhD students and postdocs.

Also, I intend to consolidate a research group, which will also include master and PhD students, in research areas having the most impact on nowadays science and engineering problems and connected with the already acquired expertise.
References


References


