University of Padova

Department of Information Engineering

Ph.D. Thesis

Calibration and 3D Mapping for Multi-sensor Inspection Tasks with Industrial Robots

Supervisor
Prof. Emanuele Menegatti
University of Padova

Co-supervisor
Prof. Alberto Pretto
University of Padova

Ph.D. Candidate
Daniele Evangelista

Academic Year
2021-2022

"Success is not final, failure is not fatal: it is the courage to continue that counts."

— Winston Churchill

Abstract

Quality inspections are an essential part of ensuring the manufacturing process runs smoothly and that the final product meets high standards. Industrial robots have emerged as a key tool in conducting quality inspections, allowing for precision and consistency in the inspection process. By utilizing advanced inspection technologies, industrial robots can detect defects and anomalies in products at a faster pace than human inspectors, improving production efficiency. With the ability to automate repetitive and tedious inspection tasks, industrial robots can also reduce the risk of human error and increase product quality. As technology continues to advance, the use of industrial robots for quality inspections is becoming more widespread across industrial sectors, ranging from automotive and manufactury to aerospace industries. The drawback of such a large variety of inspection tasks is that usually industrial inspections require specific robotic setups and appropriate sensors, making every inspection very specific and custom-built. For this reason, this thesis gives an overview of a general inspection framework that solves the problem of creating customized inspection workcells by proposing general software modules that can be easily configured to address each specific inspection scenario. In particular, this thesis is focusing on the problems of *Hand-eye Calibration*, that is the problem of accurately computing the pose, i.e., position and orientation, of the sensor in the workcell with respect to the robot frame, and *Data Mapping* that is used to map sensor data to the 3D model representation of the inspected object. For the Hand-eye Calibration we propose two techniques that accurately solve the position of the sensor in multiple robotic setups. They both consider *eye-on-base* and *eye-in-hand* robot-sensor configuration, namely, this is the way in which we discriminate if the sensor is mounted in a fixed place in the workcell or in the end-effector of the robot manipulator, respectively. Moreover, one of the main contributions of this thesis is a general hand-eye calibration approach that is also capable of handling, thanks to a unified pose-graph optimization formulation, inspection setups where multiple sensors are involved (e.g., multi-camera networks). In the end, this thesis is proposing a general method that takes advantage of a precise and accurate hand-eye calibration result to address the problem of Data Mapping for multi-purpose inspection robots. This approach has been applied in multiple inspection setups, ranging from automotive to aerospace and manufactury industry. Most of the contributions presented in this thesis are available as open-source software packages. We believe that this will foster collaboration, enable precise repeatability of our experiments, and facilitate future research on the calibration of complex industrial robotic setups.

Contents

Abstract v

List of figures ix

List of tables xiii

1 Introduction 1
- 1.1 Introduction and Main Contributions . . . 1
- 1.2 Publications . . . 3
- 1.3 Open-Source Contributions . . . 5

2 Related Works 7
- 2.1 Inspection Framework . . . 7
- 2.2 Hand-eye Calibration . . . 8
- 2.3 Data Mapping . . . 9

3 Framework for Robotic Inspections 11
- 3.1 General Overview of the Framework . . . 11
- 3.2 The offline framework . . . 12
  - 3.2.1 Process Model . . . 12
  - 3.2.2 Coverage Planner . . . 14
- 3.3 The inline framework . . . 22
  - 3.3.1 Synchronization and Data Acquisition . . . 22
  - 3.3.2 Reactive Path Planner . . . 23
  - 3.3.3 Integration of the Path Deformation . . . 25
  - 3.3.4 Extending the Offline Path . . . 27
- 3.4 Real Applications . . . 29
  - 3.4.1 Inspections for the Automotive Industry . . . 29
  - 3.4.2 Inspections for the Aerospace Industry . . . 30
  - 3.4.3 Inspections for the Manufactury Industry . . . 30

4 Local Hand-eye Calibration 35
- 4.1 Methodology . . . 36
  - 4.1.1 Eye-on-base . . . 38
  - 4.1.2 Eye-in-hand . . . 39

4.1.3 Non-linear Optimization . . . 39
4.2 Experiments and Evaluation . . . 39
4.2.1 Experimental Evaluation . . . 40
4.2.2 Real Experiments – Eye-on-base . . . 41
4.2.3 Real Experiments – Eye-in-hand . . . 43

5 General Hand-eye Calibration 45
5.1 Methodology . . . 46
5.1.1 Eye-on-Base Calibration for Single Camera . . . 46
5.1.2 Eye-on-Base Calibration for Multi-Camera . . . 49
5.2 Experiments and Evaluation . . . 52
5.2.1 Experiments with Synthetic Data . . . 52
5.2.2 Experiments on a Real Setup . . . 53

6 Data Mapping 57
6.1 Sensor Data Backprojection . . . 58
6.1.1 Backprojection Algorithm . . . 58
6.2 Image Stitching . . . 63
6.2.1 Appending a new image . . . 63
6.2.2 Calculating the 1-ring quality . . . 65
6.2.3 Template matching and Correction vectors computation . . . 66
6.3 Experimental Results . . . 66
6.3.1 Mapping with Eye-on-base Setups . . . 67
6.3.2 Mapping with Eye-in-hand Setups . . . 68

7 Conclusion 71

References 73

Acknowledgments 79

Listing of figures

3.1 The overall concept shows the software framework and its two main components: the offline framework that generates all the data needed to perform the inspection task and the inline framework that actually performs the inspection task on the robot. 12

3.2 Sketch of a perspective camera and its working range (left) and lasers with conic and pyramidal beams (right). 13

3.3 Steps of the coverage planner. 14

3.4 Voxelization. The 3D surface is sampled uniformly. For each voxel, the algorithm computes the normal of the surface (green arrow). Low-quality voxels are removed (red squares). 16

3.5 Inverse model. For a target-oriented point (red arrow), there are potentially infinite sensors poses that can be chosen to inspect it. 17

3.6 Evaluation process. 19

3.7 The validation step requires checking the reachability of the viewpoint and absence of collisions. 20

3.8 Travelling Salesman Problem: finding a route that connects all viewpoints and minimizes a given cost function. Different cost functions will lead to different routes: reduction of total inspection time for start-and-stop motion (left) or reduction of the distance between neighboring viewpoints (right). 21

3.9 (1): The path of the offline planner covers the complete area that needs to be inspected. (2): Changes in sensor positions may lead to gaps in coverage. (3): The objective of path optimization is to correct the path locally and to close the gaps. (4): The path may need to be extended to ensure full coverage. . . . 24

3.10 Workflow of the local path optimization: The blue area represents an area on the surface that was already inspected. The area covered by the current stripe is plotted in green. The grey line at the center of the current stripe is the path of the center of the sensor field of view. Each circle on this line represents one robot position along the path. The blue circles at the bottom are positions that were already sent to the robot. The grey circles belong to the path that can still be optimized. The path optimization algorithm is parametrized to work on segments (red circles) each containing 9 positions. It deforms the path towards the inspected area in order to close the gap and sends new positions to the robot on request of the robot control algorithm. In (b) the first four positions of the adapted segment were sent to the robot and the remainder is extended by four positions of the offline path. The algorithm processes this new segment until the next sending request arrives or until the segment is aligned. 26

3.11 The trajectory coming from the offline path planner is used as input for the vector field computation. The blue line represents the path of the projection center of the camera. The planned area (violet) per position is computed. A smooth vector field (black lines) is generated for the complete mesh based on the movement of the field of view of the camera on the surface. 27

3.12 Illustration of the path extension algorithm. 31

3.13 Typical engine production line examples. Courtesy of project partner *CRF*. . 32

3.14 On the left the real experimental setup developed at project partner Centro Ricerche Fiat. On the right the simulated work-cell used in the offline framework. 32

3.15 Layout of the X-Ray inspection workcell with detailed position of the X-ray source and detector and the part holders. Image courtesy of project partner FACC Gmbh. 33

3.16 On the left, pictures of the workcell implemented to test the framework with thermal inspections. On the Right pictures of the workcell implemented to test the framework with laser profilometer inspections. 33

4.1 Description of the transformations and reference frames for the two hand-eye configurations considered. 36

4.2 Result of the evaluation of the proposed hand-eye calibration method vs. Koide's method [1] (named *Graph* in the plots), Tsai's method [2], and the Daniilidis's method [3]. 40

4.3 Focus on the result of the evaluation of the proposed hand-eye calibration method vs. Koide's method [1] (named *Graph* in the plots) only. 40

4.4 Picture of the real robotic setup used for the *eye-on-base* experiments. 42

4.5 Picture of the real robotic setup used for the *eye-in-hand* experiments. 44

5.1 Multi-camera Eye-on-Base calibration setup. The goal of the proposed hand-eye calibration is to compute the position of each camera in the reference frame W, i.e., the transformations ${}^{W}T_{C1\ldots C3}$ in the figure. 47
5.2 Graphical representation of the optimization problem for eye-on-base single camera setups. The representation contains circles, rectangles and hexagons that respectively represent the estimated variables, error functions, and fixed parameters. The error functions are defined following the equations 5.3 and 5.4. 48
5.3 Evolution of the graph representation in multi cameras setups. The figure reports the error terms of Eq. (5.6-5.7) that are the stack of the errors in Eq. (5.3-5.4), respectively. Moreover, the graph includes an additional error defined in Eq. 5.10 which enforces constraints between the cameras with overlapping field of view. 54
5.4 Result of the simulated evaluation. The proposed approach, both in *single-* and *multi-camera* settings has been tested against Tsai [2], Park [4], Horaud [5], Andreff [6], Daniilidis [3], Shah [7] and Li [8]. 55
5.5 More in depth focus of the simulated evaluation regarding the proposed method only in *single-* and *multi-camera* settings. 55
5.6 Front view of the real setup where the proposed calibration method has been tested. 56
5.7 Rear view of the real setup. Main reference frames are depicted. 56

6.1 Example of the inline inspection framework GUI, visualization of the back-projection result. 60
6.2 Schematics of the two robot-sensor configurations considered for the data mapping experiments. 61
6.3 Representation of keypoints for the image stitching. A Key Point is a point of the 3D mesh that represents the objects surface. 64
6.4 Example of the ring structure used for image stitching and computation of the correction vectors of the texture to be mapped during the data mapping process. 65
6.5 Visualization of the *quality* score computed for image stitching. 66
6.6 Template matching and maximum distance of template matching. 67
6.7 Backprojection results in the *eye-on-base* experiment with thermal imaging inspection. The pictures show a visualization of the proposed software GUI within the inspection framework. Highlighted in red we have occluded surfaces (a) and detected defects (b). 68
6.8 Backprojection results in the *eye-in-hand* experiment with standard visual inspection with an industrial camera. The pictures show a visualization of the proposed software GUI within the inspection framework (a) and a detailed view of the obtained mapping results on the final textured 3D mesh (b). . . . 69

Listing of tables

4.1 Calibration results in *eye-on-base* setup. 43
4.2 Calibration results in *eye-in-hand* setup. 43

5.1 Evaluation of the reprojection errors using the real setup data. All the values in the table are in pixel unit. 51

1 Introduction

1.1 Introduction and Main Contributions

Inspection robots are increasingly being used to assess the quality of production in many industrial sectors, particularly when parts with complex geometric shapes have to be controlled. The main concept in typical inspection tasks is to move a sensor over the surface of the part so that all relevant areas are inspected. The deployment of such inspection robots proved to be difficult, time-consuming, and, consequently, expensive. One of the objectives of this thesis is to propose methodologies and software to simplify the deployment of inspection robots, for example, by contributing to the development of a software framework that enables the transition from the programming of a robotic inspection task to an easy configuration of the task itself.

This process requires generic solutions for some key challenges that take place during an industrial inspection:

- Full coverage of the part to be inspected must be ensured while avoiding collisions. Current solutions to this issue are based on manual or semi-automatic motion planning for the robot, but they proved to be excessively difficult and time consuming to implement; moreover, they require specific and dedicated procedures that change each time a new part has to be inspected or a new inspection technology needs to be involved. This coverage is particularly affected by errors in the calibration of the workcell, for example, if

the position of the sensor w.r.t. the robot is not properly measured, the inspection process may suffer from failures in the coverage of the area to be inspected, for this reason a precise calibration has to be taken in place; this process is called *hand-eye calibration*.

- To fully assess the quality of the part, a backprojection is needed. By seamlessly mapping sensor readings to the 3D model of the part, a richer 3D representation of the part can be built for further quality control. Only perfectly synchronized sensor data mapping achieves this enriched 3D model reconstruction, so accurate time synchronization must be implemented between robot movements and sensor data acquisitions.

The challenges mentioned above are the key problems addressed in this thesis work, namely the *Hand-eye Calibration* and *Data Mapping* ones. To this end, this work initially discusses and presents the formulation of a general inspection framework that considers, among other challenges, the challenges mentioned above. This software framework is the main result of an EU-funded project called SPIRIT[1] that mainly supported this Ph.D. thesis activity. In the context of the project, this thesis has developed some of the main software blocks, in particular for hand-eye calibration and data mapping, by proposing general and unified approaches that, on the one side, demonstrated enough robustness and generalization capabilities to be applied to multiple and different robotic scenarios; on the other hand, they overcome the previous state-of-the-art in terms of accuracy and efficiency. Two main contributions of this Ph.D. thesis are the hand-eye calibration methods proposed in Chapter 4 and Chapter 5. In detail, the first method is overcoming some of the limitations encountered by standard calibration approaches by proposing a local and iterative procedure to accurately compute the sensor position; instead, the last method described in Chapter 5 achieved new state-of-the-art results in calibration accuracy by proposing a general enough method that is also applicable to complex multi-camera setups. As already anticipated, another key contribution of this Ph.D. thesis is the Data Mapping approach described in Chapter 6, this method is proposing a unified procedure to perform the backprojection of sensor data over the 3D representation of the inspected object, this method can be applied, with few adaptations to different sensor's projection models such as *source-detector* (i.e., for X-Ray inspections) and standard *pinhole* (e.g., visual inspections with standard cameras) projection models.

This thesis is structured in the following way: in Chapter 2 a detailed overview of the related works for the topics addressed by this work is given. The problem of hand-eye calibration is first addressed, and the two proposed approaches are presented in Chapter 4 and Chapter 5.

[1] https://www.spirit-h2020.eu/

The Data Mapping is subsequently described in Chapter 6. Finally, the conclusions are given in Chapter 7.

The remaining sections of this chapter briefly report on the publication and open-source software development activities brought on during this Ph.D. thesis work. Some of the listed publications do not directly link with the topics addressed in this thesis, they are the result of the work done as a contributor in other research activities from the same research group of the author.

1.2 Publications

Part of this thesis has been published in peer-reviewed conferences and journal articles. Below is the list of all publications that had a central role for the finalization of this thesis work.

- **D. Evangelista**, E. Olivastri, D. Allegro, E. Menegatti and A. Pretto, "A Graph-based Optimization Framework for Hand-Eye Calibration for Multi-Camera Setups", 2023 International Conference on Robotics and Automation (ICRA), 2023. **Paper accepted**.
- **D. Evangelista**, D. Allegro, M. Terreran, A. Pretto and S. Ghidoni, "An Unified Iterative Hand-Eye Calibration Method for Eye-on-Base and Eye-in-Hand Setups", 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA), Germany, 2022.
- **D. Evangelista** et al., "Toward a Generalized Approach for Robotic Inspection Tasks". **Paper being submitted** to the Advanced Robotics International Journal.
- **D. Evangelista**, M. Terreran, A. Pretto, M. Moro, C. Ferrari and E. Menegatti, "3D Mapping of X-Ray Images in Inspections of Aerospace Parts", 2020 IEEE 25th International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna 2020. This paper has been awarded the **WIP Best Paper Award** in the category of Emerging Technologies.
- **D. Evangelista** et al., "SPIRIT - A Software Framework for the Efficient Setup of Industrial Inspection Robots", 2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT, Roma, Italy, 2020, pp. 622-626.

The following are publications that I was involved in before and during my doctorate as a collaborator, but they are not directly linked with the topics of this thesis. Nevertheless, some

of the topics and methodologies developed for this thesis may have been part of the following publications as submodules or subparts of the specific research work:

- D. Fusaro, E. Olivastri, I. Donadi, **D. Evangelista***, E. Menegatti and A. Pretto, "Pyramidal 3D Feature Fusion on Polar-Grids for Fast and Robust Traversability Analysis on CPU". Paper submitted to the International Journal of Robotics and Autonomous Systems. * **Corresponding author**.

- **D. Evangelista**, I. Donadi, D. Fusaro, E. Olivastri and A. Pretto, "Towards Accurate 3D Positioning in Large-Scale Underwater Environments", 4thItalian Conference on Robotics and Intelligent Machines (I-RIM), October 2022.

- D. Fusaro, E. Olivastri, **D. Evangelista**, P. Iob and A. Pretto, "An Hybrid Approach to Improve the Performance of Encoder-Decoder Architectures for Traversability Analysis in Urban Environments", Workshop on Online Map Validation and Road Model Creation (MaVRoC), IEEE Intelligent Vehicles Symposium (IV2022), 5th June 2022.

- D. Fusaro, E. Olivastri, **D. Evangelista**, M. Imperoli, E. Menegatti and A. Pretto, "Pushing the Limits of Learning-based Traversability Analysis for Autonomous Driving on CPU". 17th International Conference on Intelligent Autonomous Systems (IAS). 13-16 June 2022.

- A. Saviolo, M. Bonotto, **D. Evangelista**, M. Imperoli, J. Lazzaro, E. Menegatti and A. Pretto, "Learning to Segment Human Body Parts with Synthetically Trained Deep Convolutional Networks". 16th International Conference on Intelligent Autonomous Systems (IAS). 22-25 June 2021.

- M. Terreran, **D. Evangelista**, J. Lazzaro and A. Pretto, "Make It Easier: An Empirical Simplification of a Deep 3D Segmentation Network for Human Body Parts", 13th International Conference on Computer Vision Systems (ICVS), 22-24 September 2021.

- M. Terreran, L. Barcellona, **D. Evangelista** and S. Ghidoni, "Multi-view Human Parsing for Human-Robot Collaboration", 20th International Conference on Advanced Robotics (ICAR), 06-10 December 2021.

- M. Terreran, L. Barcellona, **D. Evangelista** and S. Ghidoni, "A Multi-view Framework for Human Parsing in Human-Robot Collaboration Scenarios", 3rd Italian Conference on Robotics and Intelligent Machines (I-RIM), 08-10 October 2021.

– S. Ghidoni, M. Terreran, **D. Evangelista**, C. Eitzinger, S. Zambal, E. Villagrossi, N. Pedrocchi, N. Castaman and M. Malecha, "A Smart Workcell for Human-Robot Cooperative Assembly of Carbon Fiber Parts", 3rd Italian Conference on Robotics and Intelligent Machines (I-RIM), 08-10 October 2021.

– **D. Evangelista**, M. Imperoli, E. Menegatti and A. Pretto, "FlexSight - A Flexible and Accurate System for Object Detection and Localization for Industrial Robots", 2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0&IoT), Naples, Italy, 2019, pp. 58-63.

– **D. Evangelista**, M. Imperoli, E. Menegatti, and A. Pretto, "Machine Vision for Embedded Devices: from Synthetic Object Detection to Pyramidal Stereo Matching", Austrian robotics Workshop OAGM-ARW2019.

– L. Monorchio, **D. Evangelista**, M. Imperoli and A. Pretto, "Learning from Successes and Failures to Grasp Objects with a Vacuum Gripper", Task-Informed Grasping (TIG) for rigid and deformable object manipulation (IROS 2018 Workshop).

– **D. Evangelista**, W. U. Villa, M. Imperoli, A. Vanzo, L. Iocchi, D. Nardi and A. Pretto, "Grounding Natural Language Instructions in Industrial Robotics", Human-Robot Interaction in Collaborative Manufacturing Environments (IROS 2017 Workshop).

– **D. Evangelista**, F. Iodice, A. Perica, M. Cefalo, E. Magrini, M. Anzidei and M. Venditelli, "Residual-based interaction force estimation for haptic feedback in teleoperated needle insertion", accepted for poster session at CRAS2016, Computer Robot Assisted Surgery, held in Pisa on September 2016.

1.3 Open-Source Contributions

Some of the algorithms and techniques presented in this thesis are also supported by open-source software implementation to facilitate future research:

– Chapter 5 presents our proposed graph-based general hand-eye calibration method. The implementation is available at: `https://bitbucket.org/freelist/gm_handeye`

– The author of this thesis also contributed to the development of the work presented in [9]. The implementation is available at: `https://bitbucket.org/flexsight/traversabilityanalysis`

2
Related Works

This chapter provides an extended overview of the related works for all the main topics addressed by this thesis work. In particular, an initial overview of the state-of-the-art approaches for industrial inspections will be given in Sec. 2.1. After this review, in Sec. 2.2 and Sec. 2.3, an extended description of the related works for the *hand-eye calibration* and *data mapping* tasks will be given, respectively.

The works presented in the following sections represent the backbone of the research study addressed during the initial phases of each of the main topics addressed by this thesis work; in particular, for the hand-eye calibration, many of the state-of-the-art methods presented in Sec. 2.2 have also been used for comparison with the two proposed calibration approaches in Chapter 4 and Chapter 5 respectively.

2.1 Inspection Framework

Quality inspection in industry is a task that has been often demanded to humans in the past [10]. Even if in many applications humans can achieve higher performance compared to machines, they are slower and get tired quickly. Moreover, humans require training in industry and most of the time this is an expensive and time consuming process. Furthermore, in many industrial environments (e.g. nuclear industry, chemical industry, radioactive environments, etc.) inspection may be dangerous for human operators. For these and many other reasons, machines and computer vision have intensely replaced humans in such scenarios [11]. Although

quality inspection based on machine and computer vision systems is nowadays a quite common process that is performed in many industrial sectors, it requires a specific and tailored configuration of such systems, and both hardware and software specifications are different depending on the specific use case. For example, in the food industry quality inspection is mainly done visually, using visual features such as color or shape [12] [13]. Photometric technology is used instead for inspecting composite parts and carbon fibers [14].

The inspection technology itself is not only the single source of entropy in such an etherogeneous sector of the industry. In fact, since quality inspection also involves robots and automated systems, each specific application requires an appropriate robotic solution that spread over common industrial manipulators such as for the inspection of structural elements in construction industry [15] to UAVs (Unmanned Aerial Veichles) and UGVs (Unmanned Ground Veichles) in the agriculture industry [16].

All the aforementioned examples require specific hardware and software, and specific configurations and required training for usage that make almost impossible the scalability of such systems to more a generic and standardized approach. To this end, the SPIRIT project aims to propose a general software framework that can be easily re-adapted and re-configured for the specific inspection task with few effort for the human operator. More in particular, it will be shown how this framework has been used in multiple industrial sectors.

2.2 Hand-eye Calibration

In the literature, several methods have been proposed for hand-eye calibration. Shiu and Ahmad [17] initially proposed a method for the estimation of the hand-eye transformation based on the solution of a homogeneous equation. Tsai and Lenz [18] proposed a widely adopted calibration method (e.g., implemented in [19]) that first estimates the translation and then the rotation of a hand-eye transformation. Similarly to Shiu and Tsai, Chou and Kamel [20] solved an estimation problem based on a normalized quaternion representation to transform the kinematic equation into two simple and structured linear systems with rank-deficient coefficient matrices. Daniilidis and Bayro-Corrochano [3] proposed a similar framework that is based on dual quaternion parameterization and singular value decomposition, and Srobl and Hirzinger [21] introduced a novel metric to optimize the estimation problem by appropriately weighting translation and rotation errors in the optimization problem. Park and Martin [4] estimated the hand-eye transformation in the Euclidean group using Lie group and least-square optimization. Gwak *et al.* [22] proposed a cyclic coordinate descent algorithm to optimize

objective functions in SE(3) and applied it to a class of robotic problems such as hand-eye calibration. In this direction, Horaud and Dornaika [5] proposed a linear formulation of the same optimization problem. The same method has been further applied to online calibration by Andreff *et al.* [6].

More recently Shah [7] formulated a closed-form solution for the hand-eye problem by using an SVD-based algorithm and the Kronecker product to solve for rotation and translation separately, while LI *et al.* [8] instead used dual quaternions to solve them simultaneously to overcome the limitations of the Kronecker product.

More general formulations of the hand-eye calibration problem have been done in [23] and [24]. The latter, in particular, mathematically formulates the problem of camera-to-camera calibration using the geometric constraints of hand-eye calibration when a common reference frame (e.g., robot base frame) is taken into the loop. Although this method formally extends the formulation of [18] to multi-camera setups, it requires an external motion capture system to accurately recover the position of cameras during calibration, making the whole approach not easily scalable or applicable to small setups.

In [1], they proposed a more general approach based on pose graph optimization that achieves high levels of accuracy and introduces a general solution that can be easily applied to different robotic setups and camera projection models (e.g., both standard pinhole camera projection and X-ray source detector models). Although the method proposed in [1] is quite general, it does not cover eye-on-base setups in single-camera settings, and it does not handle multi-camera robotic systems.

In this thesis, we overcome this limitation by extending the approach to eye-on-base setups. We initially proposed a local and iterative method that handles both *eye-on-base* and *eye-in-hand* setups, See Chapter 4. Moreover, we further extended all the previous approaches giving a more general formulation applicable also to multi-camera setups where the optimization problem has been shaped as a graph-based robust optimization problem, See Chapter 5.

2.3 Data Mapping

Industrial inspection is nowadays a common task performed in many sectors using machine vision approaches and different hardware depending on the specific industrial scenario.

In automotive and aerospace industry, normal cameras are not suitable for quality inspection of composite parts and carbon fibers, due to the challenging optical properties of such materials. In such cases common solutions are infrared cameras [25], X-ray sensors [26][27] or specific

sensors designed to measure fibre orientation [28].

In wood processing industry, X-ray computed tomography (CT) technology is used for non-invasive assessment of log internal feature, as the geometry and position of knots [29] [30]. In this case, a rotating X-ray source produces cross-sectional images that can be used to generate three-dimensional (3D) reconstructions of the scanned object including information about the outer shape, internal knottiness and identified defects.

Although CT technology outputs very detailed 3D reconstructions, it requires that the part to be inspected is smaller than the CT machine. This could be not possible or unfeasible in particular inspection task, for example when the part is too large or it has a peculiar shape.

In this scenario, an alternative approach is X-ray stereo, which uses an X-ray source and a detector for image acquisition [31]. The 3D reconstruction is then obtained with photogrammetric methods combining X-ray images from different point of views.

The inspection process can be further automated using an industrial robotic arm to move a sensor along the part to be inspected while acquiring data.

In [32] a specific sensor to measure fibre orientation is mounted on a robot arm. Using such robot arm, the sensor is moved on a predefined path to inspect a carbon fibre part, namely its fibre orientations. The data acquired are then mapped onto the CAD model to obtain a fully 3D representation of the part's fibre angles.

In this thesis, we are proposing a data mapping approach that adopts a similar strategy. Sensor data is directly mapped onto the surface of the CAD model of the part to be inspected, to obtain a textured CAD model where defects can be easily identified. One of the key advance of the proposed approach, in particular for the field of X-ray inspections. is that, to our knowledge, this kind of approach has not been applied in a real manufacturing context before. The proposed approach has been presented in [33] and got the *Best WiP Paper Award* at the *International Conference on Emerging Technologies and Factory Automation (ETFA)* on 2020. See Chapter 6 for further details.

3

Framework for Robotic Inspections

3.1 General Overview of the Framework

In Fig. 3.1 the main concept of the proposed framework is shown. In particular, given a 3D CAD model of the part that represents the object to be inspected, a planning step is performed first. At this stage, the information about the sensor physical model and the shape of desired parts to be inspected are merged to produce a consistent path that is at the same time collision-free and also feasible for the involved robot manipulator. The generated path is then sent to the inline framework, which is responsible for its reproduction in the real robotic setup. During the online process, sensor data and robot positions are mutually synchronized and reactive behaviors are applied to obtain effective and consistent results during the last step of data back-projection from sensor acquisition to the 3D object model. Then, this last step is entered into an evaluation module that is responsible for accepting or rejecting the part. It is important to note that the inspection method itself is not the focus of the framework, since the idea behind the proposed method is to implement a generic approach capable of dealing with each inspection process, without focusing attention on the specific inspection technology, sensor type, or robot model.

In the following sections, the two main parts of the framework will be described in more detail.

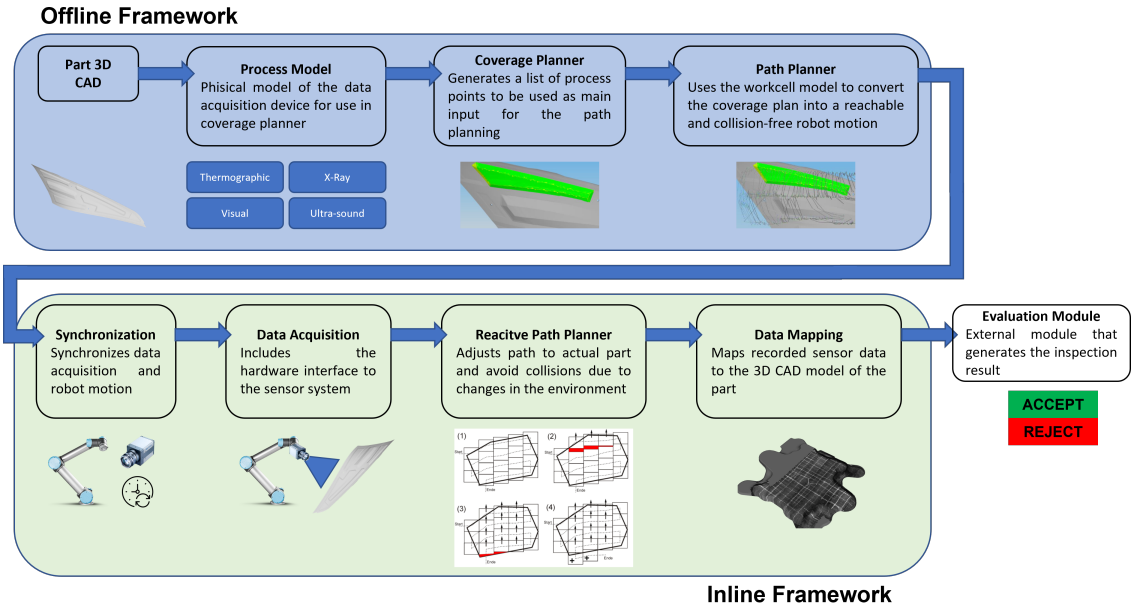


Figure 3.1: The overall concept shows the software framework and its two main components: the offline framework that generates all the data needed to perform the inspection task and the inline framework that actually performs the inspection task on the robot.

3.2 The offline framework

3.2.1 Process Model

To produce a consistent robot trajectory, it is needed to know what is the exact physical representation of the sensor involved during the inspection, namely what are the characteristics of the image acquisition device, e.g. camera field of view, focal lengths, principal point, etc. All these features are provided within our offline framework through what we call *Process Model*, with which the user can directly specify each sensor property or specific feature and how the inspection will be performed. This process model has been specified to include a wide range of inspection technologies, e.g.: standard 2D cameras, 3D sensors, X-Ray source/detector devices, thermal cameras, laser profilometers.

In the following, we will give a detailed explanation of how we model and parametrize an inspection tool within the proposed *Process Model*. An inspection tool consists of a set of components that can be either cameras or light projectors such as lasers or X-Ray sources. In this work, two types of cameras have been modeled: orthographic camera for the X-ray source/detector inspection tool and perspective cameras for all the other inspection tools. To define a

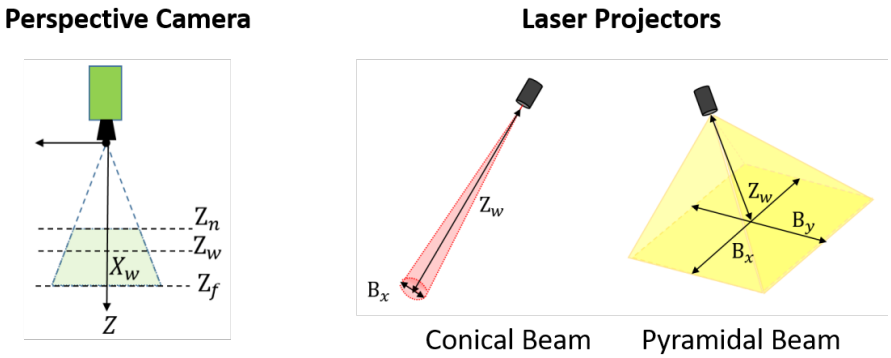


Figure 3.2: Sketch of a perspective camera and its working range (left) and lasers with conic and pyramidal beams (right).

common camera model, we identified two sets of parameters that represent spatial and temporal constraints, respectively. The spatial parameters are: pixel resolution $(R_x \times R_y)$, pixel size s and working distance Z_w. The temporal parameters are: exposure time T_{exp} and maximum frame rate T_{fmax}.

In general, X-Ray sensors can be handled using only these parameters; instead, standard perspective cameras also require information about the focal length L and the lens aperture (f-stop) f. These parameters are required to calculate the near plane Z_n and the far plane Z_f that delimit the depth of the field, see Fig. 3.2.
If we consider *Circle of Confusion (CoC)* to be equal to the pixel size s, the depth of the field can be computed as follows:

$$H = \frac{L^2}{f * s} \tag{3.1}$$

$$Z_n = \frac{H * Z_w}{H + (Z_w - L)} \tag{3.2}$$

$$Z_f = \frac{H * Z_w}{H - (Z_w - L)} \tag{3.3}$$

where Eq. Eq. (3.1), Eq. (3.2) and Eq. (3.3) are the computations of the hyper-focal distance, near plane and far plane, respectively. The last information required to compute the region observed by the camera is the field of view. At the working distance, *horizontal field of view* X_w

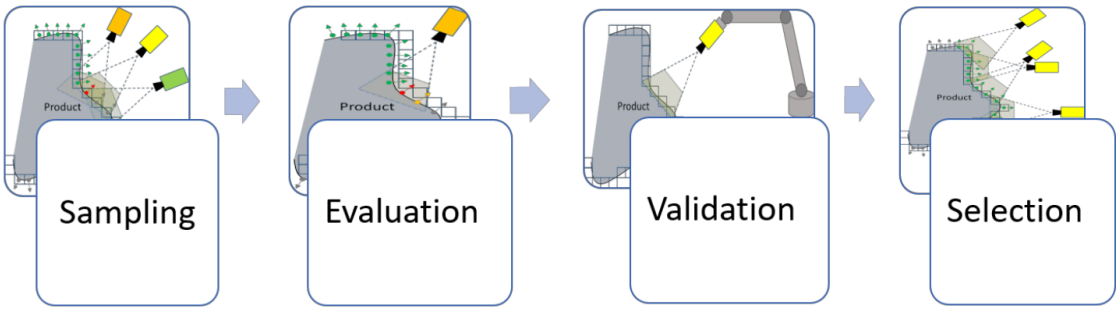


Figure 3.3: Steps of the coverage planner.

and *vertical field of view* Y_w are calculated as follows:

$$X_w = \frac{s * R_x * Z_w}{L} \tag{3.4}$$

$$Y_w = \frac{s * R_y * Z_w}{L} \tag{3.5}$$

Following the same concept that we used for modeling cameras, light projectors can be defined by their beam shape. Two beam shapes, that is, pyramidal and conical (see Fig. 3.2), have been found to be enough to handle most use cases in industrial scenarios. For example, the conical shape is used to model both the laser of a thermal camera and the X-Ray source, while the pyramidal shape is used to model the light pattern of a 3D sensor or the laser line emitted by standard profilometers. The parameters of each lighting tool are the following:

- Aperture of the laser beam defined using angles (B_x, B_y) in the X and Y directions for the pyramidal beam and the radius for the conical beam;
- Projection depth Z_w, which together with the aperture are used to compute the base of the irradiating light.

3.2.2 Coverage Planner

The key element of the offline framework is a simulated environment that emulates the real work cell by accurately reproducing the physical model of the sensor and the kinematic model of the involved robots. This simulator is an evolution of the work done in [34]. The most important modules of this simulator are the Coverage and Path Planners. In detail, the cover-

age planning module is in charge of finding the viewpoints required to completely cover the inspection surface of the product.

In this context, a viewpoint is defined as the position of the sensor in the product-centered reference frame. As shown in Fig. 3.3, the coverage planning consists mainly of four steps: *sampling, evaluation, validation* and *selection* for each viewpoint.

- *Viewpoint Sampling*. It consists of sampling the surface of the inspected product. For each sampled point on the surface, an inverse model of the inspection tool is used to generate a bunch of viewpoints from which the sampled point is visible. Since the inverse model provides several solutions, a heuristic method to test the best (in terms of imaging quality) configuration first is applied.

- *Viewpoint Evaluation*. It is a procedure to measure how good a viewpoint is with respect to the others. In the proposed framework, the quality of a viewpoint depends on the number of visible meshing triangles and on how much the projection of the inspected area is centered on the sensor surface. A threshold on viewpoint quality is used to discard viewpoints that do not provide enough information. The evaluation is performed by using the forward model of the sensor, which provides a unique solution because the visible surface is uniquely determined by the position of the product with respect to the sensor and by the shape of the product itself.

- *Viewpoint Validation*. In order to be added to the coverage plan, a viewpoint must be valid. In this context, a viewpoint is valid if the robot is able to move the product or the sensor to the required position and the target robot configuration is collision free; that is neither the robot nor the sensor touches other entities of the work cell. In this stage, the inverse kinematics of the robot model is used to check whether the robot can reach the target position. This step requires knowing the size of the robot links and the limits of joint movements.

- *Viewpoint Selection*. The selection procedure is in charge of selecting the viewpoints from the valid ones in order to fulfill the inspection constraints. The authors identified the following constraints for this stage: the most important inspection constraint is that the selected viewpoints should provide a complete coverage of the inspection area; the second constraint tries to keep the amount of overlap among neighbor viewpoints close to the desired one; finally, the coverage plan tries to maximize the overall goodness of

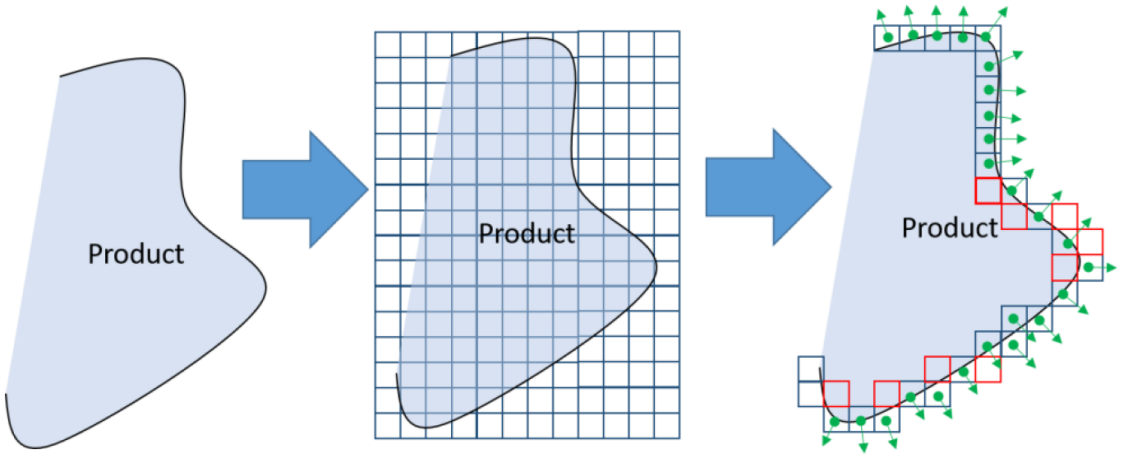


Figure 3.4: Voxelization. The 3D surface is sampled uniformly. For each voxel, the algorithm computes the normal of the surface (green arrow). Low-quality voxels are removed (red squares).

the selected viewpoints, in particular, viewpoints that observe the inspection area at the center of the working region are preferred.

In the following sections, the coverage planning steps aforementioned are described in detail.

Viewpoint Sampling and Inverse Model of the Sensor

The product to be inspected is represented within the simulator using a triangle mesh representation, i.e. a set of triangles that are connected by their common vertices. The inspection surface is represented by a list of indexes corresponding to the selected triangles of the mesh. Although a mesh representation is useful for visualization, it is not suitable for a uniform sampling of the inspection surface. To achieve uniform sampling of the surface, the mesh needs to be voxelized. As summarized in Fig. 3.4, the surface of the product is sampled on a uniform 3D lattice to obtain small squared boxes, called voxels. A normal vector is assigned to each voxel by means of the set of triangles that intersect the voxel. The intersection areas of the triangles and a voxel are used as weights, and the weighted sum of the triangle results in the normal vector of a voxel. The advantages of using voxels instead of meshes are the following: uniform sampling of the space; richer representation of the surface normal due to mesh averaging; simple representation of the connections among voxels. Once the surface has been sampled, the voxels are used to generate viewpoints that can potentially be added to the coverage plan. The idea is to generate viewpoints that can be used to inspect the sampled point on the surface. However, since there could be an infinite number of viewpoints that cover the sampled point, some

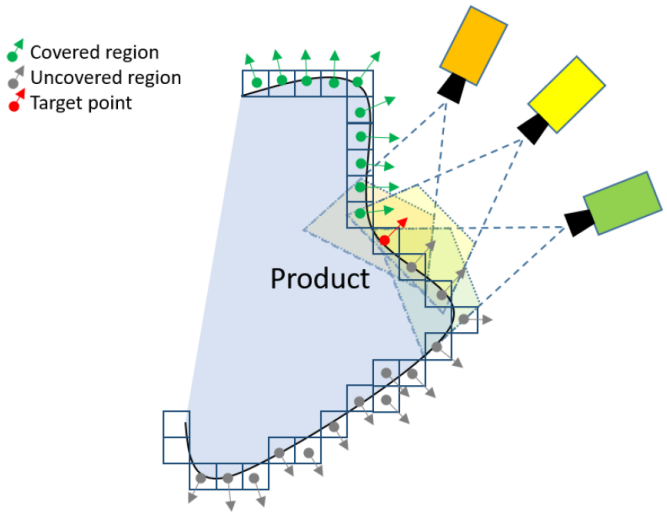


Figure 3.5: Inverse model. For a target-oriented point (red arrow), there are potentially infinite sensors poses that can be chosen to inspect it.

heuristic is required to limit the number of solutions. The implemented heuristic depends on the selected inspection tool and its inverse model (see Fig. 3.5 for reference). For a standard camera, the implemented inverse model picks the viewpoints so that the sampled voxel is in the center of the field of view and is perfectly focused. For inspection tools comprising a camera with a laser, such as the thermo-camera and profilometers, the laser is placed so that the voxel is in the center of the laser beam and the laser ray is parallel to the surface normal. The distance between the surface and the sensor is chosen so that the sampled voxel is in working distance (in focus). For the X-ray technology, the X-ray source is placed perpendicular to the surface, and the sensor is placed at a desired distance on the other side of the product.

Among the selected viewpoints, those that have their optical axes parallel to the normal of the voxel will have higher priority. If this configuration is occluded (see the viewpoint evaluation description in the next section), viewpoints are tilted until a valid solution is found, or the inclination between the optical axis and the voxel normal is higher than the maximum allowed tilt angle, which is a parameter of the inspection process. Tilting the sensor has been found to be better than changing the working distance because usually the depth of field of the inspection sensor is quite narrow and it is unlikely that a valid viewpoint can be found by changing the distance between the surface and the sensor.

The above-described sampling procedure provides a lot of viewpoints for each voxel. The coverage planning algorithm needs to select some of them to cover the whole inspection area while maintaining a certain amount of overlap. Thus, the selection process requires knowing which triangles of the mesh are seen by each viewpoint. The procedure of computing the visibility from a point of view is known as the *forward model* of the inspection tool. Two methods have been developed to compute the forward model of the inspection tools, one for cameras coupled to lighting tools (e.g. 3D sensors and the thermal cameras) and another for the X-Ray inspection. In both cases, instead of using ray tracing [35], an exhaustive search has been carried out on each triangle of the mesh to check whether its centroid is visible by the sensor or not. For a camera with light projectors, this computation is performed following the approach proposed in [36]. Briefly, a point is visible if it is irradiated by at least one light projector and if it falls within the working range of the camera. This means that the following constraints should be verified: the centroid of the triangle is in the field of view of the camera; the line between the optical center of the camera and the centroid is not occluded by other parts of the product; the angle between the surface normal and the optical axis is lower than a threshold (angular range); the centroid of the triangle is inside the beam of at least one lightening tool (shape and distance will be considered); the line between the lightening source considered above and the centroid of the triangle is not occluded by other parts of the product; the angular range between the surface normal and lightening source is lower than a threshold.

Moreover, for X-Ray inspections, a triangle will be considered as inspected if: it is intersected by at least one ray originating from the X-Ray source and hitting the X-Ray sensor; if required, we may consider adding constraints about 1) the distance between the triangle and the source and 2) the angular range between the ray and the surface normal.

Calculating which triangles of the mesh can be inspected from a given viewpoint is the first step of the evaluation process. The second step is to judge the quality of the viewpoint. As depicted in Fig. 3.6, the quality of a viewpoint depends on the number of new observed triangles, the overlap with other viewpoints already included in the coverage plan, and how much the inspected region is centered. The quality of the viewpoint is used by the viewpoint selection algorithm described in the next section to choose the best viewpoint in a local region of the mesh to update the coverage plan.

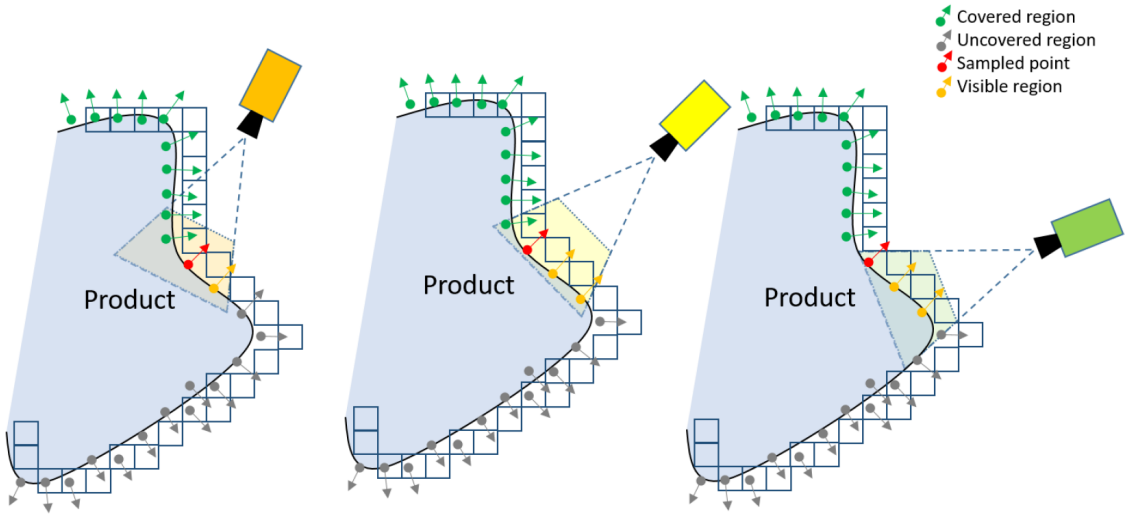


Figure 3.6: Evaluation process.

Viewpoint Validation

Viewpoints provide information about the position of the sensor with respect to the product. The role of the robot is placing the sensor at the desired viewpoint location while avoiding collisions with the cell. Depending on the type of cell, the robot can act as a manipulator, i.e. it must move the part in front of the sensor, or as an inspector, which moves the sensor in front of the product. As depicted in Fig. 3.7, the validation step consists of checking the reachability of the viewpoint and ensuring the absence of collisions among robot, sensors, product and cell.

The validation process comprises the following steps. First, the viewpoint location is used to compute the required position of the robot-flange in the robot-base coordinate-system. Then, the inverse kinematic solver computes the associated robot configurations taking care of the different sensor-robot possible configurations: *eye-in-hand* or *eye-on-base*.

Viewpoint Selection

This section describes how the coverage planner takes the input from the previous steps in order to select the viewpoints to be added to the plan W. The algorithm has the following steps:

1. Find a starting voxel (a corner in the selected product) and add it to the queue Q;
2. Take the first point q from the queue Q and remove it from the queue;
3. Selection: find the best viewpoint w that sees q;

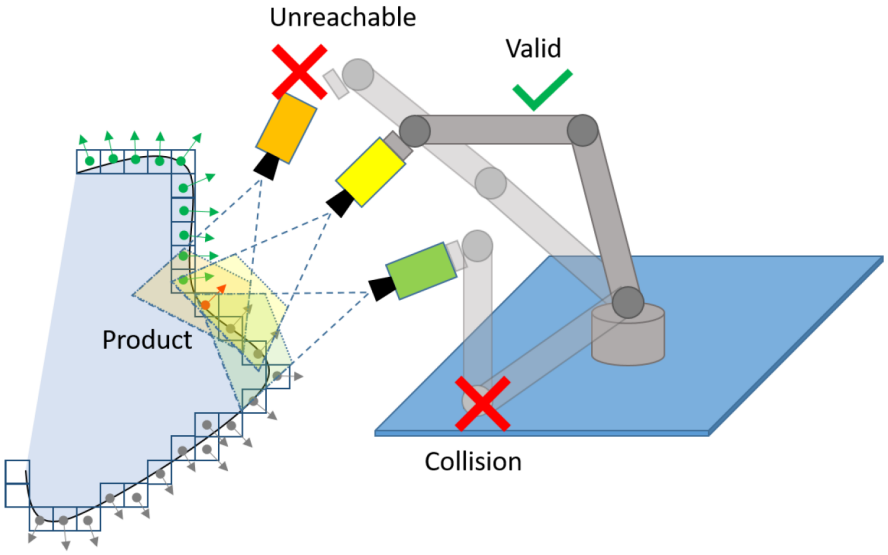


Figure 3.7: The validation step requires checking the reachability of the viewpoint and absence of collisions.

4. If w is good enough add it to the selected viewpoints W;
5. Find the contour of w on the part, defined as the closest voxels to q that are not seen by w;
6. Get from the border of the areas observed by w two points q_1, q_2 that have not been inspected yet and (if any) add them to Q;
7. If Q is empty continue; else go back to 2);
8. Return W.

Step 3 of the algorithm represents the selection method, which searches the best viewpoint that sees q. This is accomplished by generating viewpoints from the neighbourhood of the input voxel q using the inverse model of the inspection sensors described in previous sections. A viewpoint v is selected as the best viewpoint if it fulfils three conditions:

- it sees the input voxel q. This condition is verified using the forward model of the sensor;
- it is valid. This condition is verified using the viewpoint validation strategy;
- It is better than the other viewpoints. As described in previous sections, the evaluation compares the size of the regions inspected by the viewpoints and how much they are centered w.r.t sensor image plane.

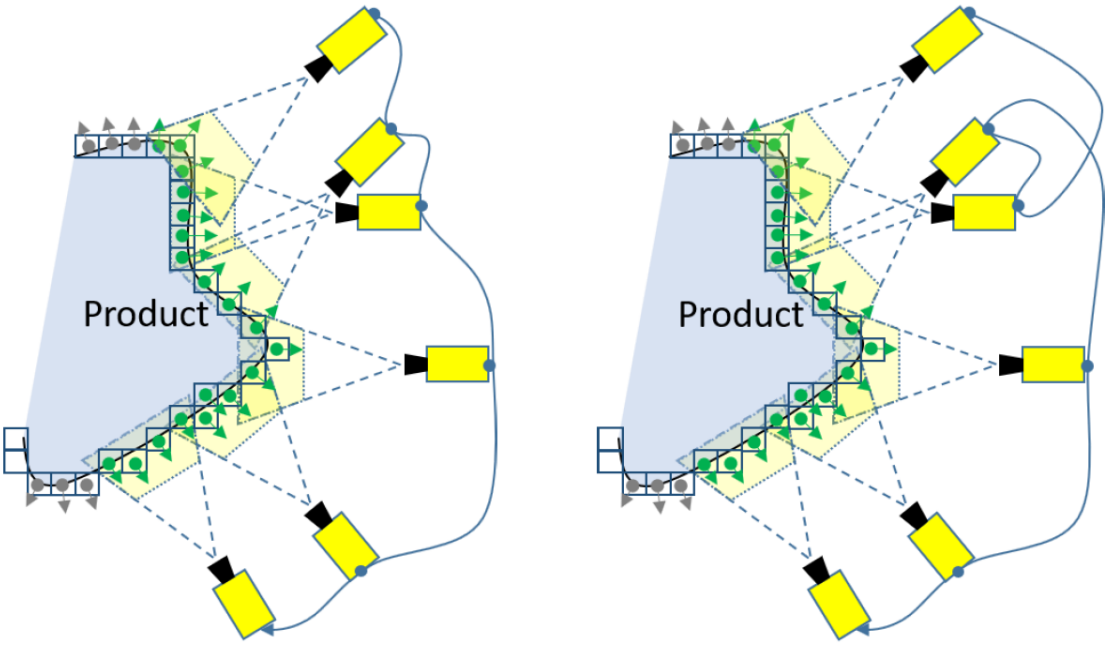


Figure 3.8: Travelling Salesman Problem: finding a route that connects all viewpoints and minimizes a given cost function. Different cost functions will lead to different routes: reduction of total inspection time for start-and-stop motion (left) or reduction of the distance between neighboring viewpoints (right).

In details, the selection algorithm works as follows:

1. Create a queue S containing the input viewpoint q;
2. Take the first point s from the queue S and remove it from the queue;
3. Use the inverse model of the sensor to get a viewpoint v that sees s;
4. If the viewpoint v sees q, v *is better than* w and v is valid then set $w = v$;
5. Add the voxels in neighbourhood of q that are seen by v to S;
6. If S is empty continue; else go to 2).

The viewpoints W generated by the coverage plan are then sent to the path planning algorithm, which is in charges of finding a path to pass through them avoiding collision and reducing the duration of inspection process.

3.3 The inline framework

Once the robot trajectory has been generated so that specific constraints are satisfied and desired inspection areas are fully covered, the inline framework is responsible for reproducing what has been planned before hand. To effectively solve the inspection task, robot motion and data acquisition must be synchronized; this ensures correct execution of the inspection, and each sensor frame can be further used to perform consistent data mapping on the part 3D model representation.

3.3.1 Synchronization and Data Acquisition

The synchronization of robot motion and sensor acquisition has a specific and central role in the in-line framework for inspections. It is handled by a specific module, developed in [37]. This module requires real-time communication with the hardware interfaces of the robot and sensor, it is based on reprojection error minimization and the core of the method is resumed below.

The goal of this module is to compute the time offset Δt that exists between robot motion and sensor acquisition. This time offset is estimated by comparing sequences of non-synchronized robot hand poses and camera images (i.e., in the specific case of imaging sensors). To compare them, the robot hand is moved along a certain path (e.g., predefined trajectory above the calibration pattern), for each pose along the path images and robot hand (R, t) transforms are stored. Let $\Re$ be the robot hand pose sequence, and I be the set of images in the sequence. Let us now assume that the total numbers of acquired images and poses are N and K respectively, and the calibration pattern consists of M points.

Let $\hat{p}_t^i$ be the i-th point of the calibration pattern seen by the j-th image at timestamp $\hat{t}$. Given a robot-camera time offset Δt, we compute the robot hand pose R_t at the corresponding robot time $t = \hat{t} + \Delta t$ by interpolating the discrete robot hand poses $\Re$. We used the spherical linear interpolation (*Slerp*) [38] to interpolate the robot hand poses:

$$R_t = Slerp(R, t) \tag{3.6}$$

Using the interpolated robot hand pose R_t we are able to project each point of the calibration pattern p^i onto the camera image plane and then compute the reprojection error defined as

follows:

$$E(\Delta_t) = \sum_{\hat{t}}^{N} \sum_{i}^{M} \parallel \hat{p}_t^i - Porj(Slerp(R, \hat{t} + \Delta_t), p^i) \parallel \tag{3.7}$$

Thus, minimizing the reprojection error in Fig. 3.7 we are able to estimate the robot-camera time offset $\Delta\hat{t}$:

$$\Delta\hat{t} = \arg\min_{\Delta_t} E(\Delta_t) \tag{3.8}$$

3.3.2 Reactive Path Planner

During each inspection trial, the offline computed path must be adjusted considering possible changes in the environment due to actual part differences compared to its 3D CAD model that may generate collisions that cannot be precomputed during the off-line stage. For this purpose, the proposed inline framework includes a *Reactive Path Planner* module that corrects "on the fly" the given path by detecting such changes or defects on the surface of the component. The requirement for local path optimization arises primarily if the area that a sensor inspects differs from the planned area, and this *inconsistency* typically arises when one or both of the following cases happen:

1. The sensor position is changed based on a Cartesian correction;

2. During the evaluation of the sensor data, it turns out that some areas that are planned for the current sensor position, cannot be evaluated due to insufficient data quality. Typical example of this is with laser profilometer sensors when, depending on reflection properties of the scanned surface and on its curvature, the intensity of the reflected laser light might be too weak to be detected by the camera. As a result, the width of the stripe of inspected surface area is narrower than assumed by the offline coverage-planning algorithm.

Changes in sensor coverage might lead to unnecessarily long inspection times in regions where the overlap between neighboring sensor positions increases. The most problematic case occurs if the overlap between sensor positions is decreased until gaps in the coverage plan appear, an example is given in Fig. 3.9. To cope with this issue, a specific local path optimization

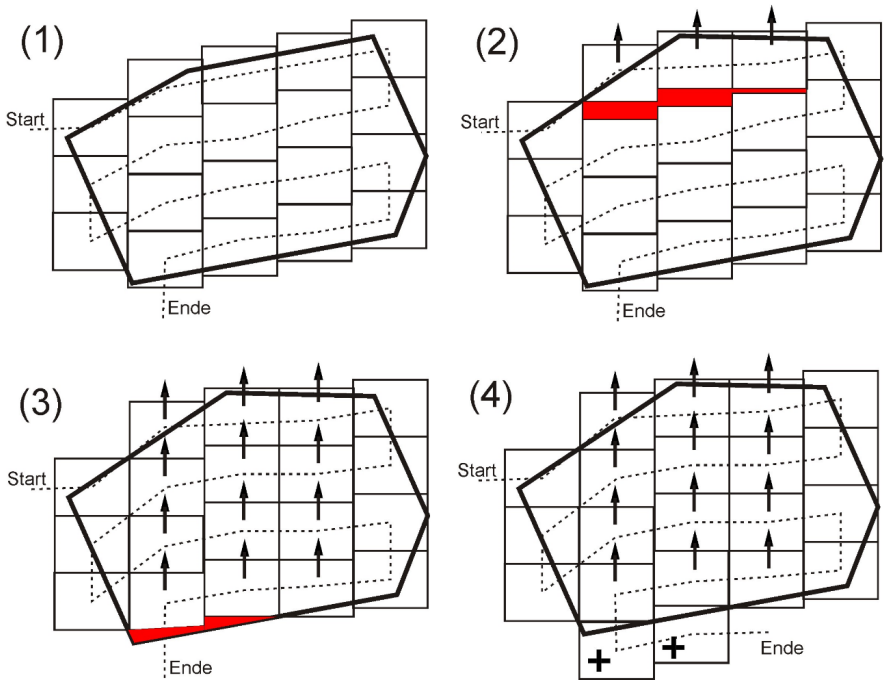


Figure 3.9: (1): The path of the offline planner covers the complete area that needs to be inspected. (2): Changes in sensor positions may lead to gaps in coverage. (3): The objective of path optimization is to correct the path locally and to close the gaps. (4): The path may need to be extended to ensure full coverage.

algorithm that aims at correcting for gaps and large overlaps in the acquired data patches by re-alignment of the sensor positions has been developed and its main consteps can be summarized as follows:

- The robot control block and the sensor data evaluation block provide input to the algorithm. The robot control sends segments of the trajectory that need to be optimized, and the data evaluation block sends information about the areas on the surface that were inspected.
- Areas of the surface that are already inspected are ignored by the path optimization. From the perspective of the algorithm, the mesh is continuously shrinking and the goal is to align the current path segment with the border of the mesh.
- The optimization starts when a new path segment is received and the algorithm iteratively deforms the trajectory segment by performing the following steps:

    1. Correction vectors are computed for all positions in the segment;

2. The segment is deformed by means of the correction vectors and the planned area is updated;
3. The algorithm stops if the optimization converges or when the time for computing further iterations is exceeded. Otherwise, Step 1 is performed again.

During the execution of the path, while trajectory points are corrected as described above, data are continuously mapped onto the 3D CAD model of the part using the method developed and proposed in [33].

3.3.3 Integration of the Path Deformation

The local path optimization algorithm that was described in the above Section iteratively adapts a path by computing correction vectors that move the positions of the path in Cartesian space. The trajectory is slightly deformed at each iteration and the optimization stops when the trajectory is aligned, i.e. when the size of the deformation falls below a threshold.

Some issues need to be considered, when the local path optimization algorithm is combined with the reactive path planning:

- Only a short segment close to the current robot position of the trajectory can be optimized since the data of the sensor provide only local information.
- Positions that were sent to the robot cannot be changed anymore. The local path optimization needs to consider these positions to guarantee that the trajectory always stays continuous and smooth.
- The path deformation is done in Cartesian space and there might occur problems related to the robot kinematics, when the deformed trajectory is transformed back into joint space, e.g., there is no solution to the inverse kinematics, or the robot would collide with an object in the work-cell. There must always be a fall-back strategy, which prevents the robot from moving into a dead end.

To tackle these problems, the following strategy can be implemented:

- The local path optimization always only adapts a short path segment starting at the last position that was sent to the robot (see Fig. 3.10).

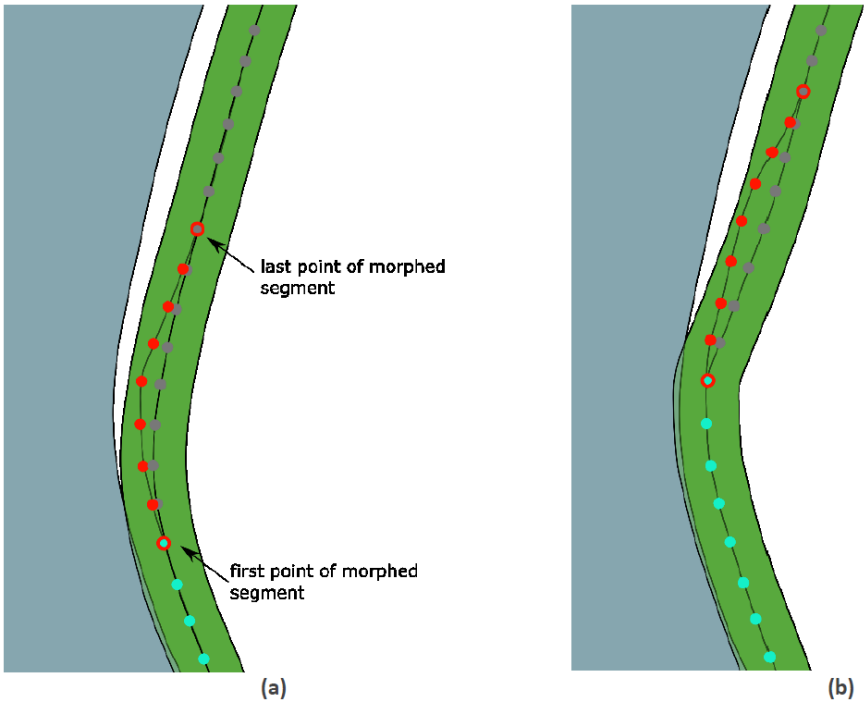


Figure 3.10: Workflow of the local path optimization: The blue area represents an area on the surface that was already inspected. The area covered by the current stripe is plotted in green. The grey line at the center of the current stripe is the path of the center of the sensor field of view. Each circle on this line represents one robot position along the path. The blue circles at the bottom are positions that were already sent to the robot. The grey circles belong to the path that can still be optimized. The path optimization algorithm is parametrized to work on segments (red circles) each containing 9 positions. It deforms the path towards the inspected area in order to close the gap and sends new positions to the robot on request of the robot control algorithm. In (b) the first four positions of the adapted segment were sent to the robot and the remainder is extended by four positions of the offline path. The algorithm processes this new segment until the next sending request arrives or until the segment is aligned.

- The first point of the segment is the last point that was sent to the robot. It is fixed during the optimization since it cannot be changed anymore, but it ensures that the robot can always smoothly follow the morphed trajectory.

- The last point of the optimized segment is also fixed. As a result, it is ensured that the optimized segment is always smoothly connected to the original trajectory. This prevents the robot from running into a dead end: The original trajectory is admissible regarding the robot kinematics and collisions. After each iteration of the path optimization, we check if the new segment is still valid. The new segment is rejected in case there is no collision-free solution to the inverse kinematics. In this case its predecessor segment continues the robot motion. Even if it is not possible to perform a single valid path op-

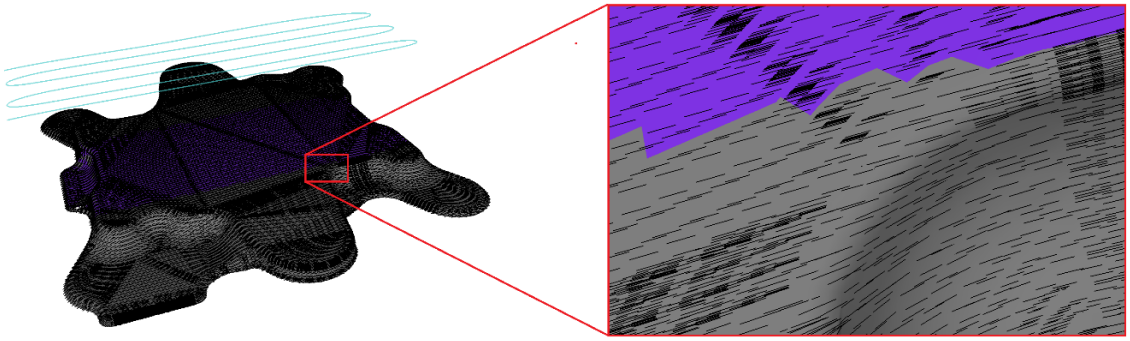


Figure 3.11: The trajectory coming from the offline path planner is used as input for the vector field computation. The blue line represents the path of the projection center of the camera. The planned area (violet) per position is computed. A smooth vector field (black lines) is generated for the complete mesh based on the movement of the field of view of the camera on the surface.

timization iteration, it is always guaranteed that the current segment is admissible and leads back to the original trajectory, which is also valid. Hence, the robot will never get stuck and there is always a smooth extension to its current position.

3.3.4 Extending the Offline Path

The path optimization described above locally deforms the offline trajectory in order to optimize the coverage. However, full coverage might be lost, if the mismatch between the simulated coverage of the offline path planning stage and actual covered areas is too large. In this case, the inline framework tries to close the remaining holes by extending the offline path and adjusting in the proper way the robot trajectory.

Computation of the Movement Strategy

At the heart of the path extension algorithm is a vector field (see Fig. 3.11), that serves as a movement strategy for the extension of the offline path. It consists of a direction for each triangle of the mesh and is derived from the offline trajectory as follows:

1. The process model is used to compute the planned areas on the surface at each position of the trajectory.

2. The trajectory positions are ranked according to their associated planned area. At the best positions (large areas) we compute the direction of motion field of view of the sensor. At each selected position, the centers of the field of view and the motion direction are projected onto the mesh. Each projection hits a certain triangle of the mesh.

3. By means of these pairs of triangles and directions, we use the algorithm described in [39] to compute a smooth vector field for the complete mesh.

Computation of a Path Extension

Based on the vector field a new trajectory stripe can be computed by means of the following iterative algorithm (see Fig. 3.12):

- Mark all triangles as valid that were planned by the offline planner, but that remained un-inspected.
- Choose one of the valid triangles that needs to be inspected. Its center of mass serves as the seed for a path on the surface of the mesh.
- Try to extend the surface path at both ends by integrating the vector field/ respectively the inverse of the vector field.
- Compute for each surface position a sensor position by means of the process model.
- The inverse kinematics are applied to all Cartesian sensor positions. All positions that do not have a collision-free solution are removed from the new segment. In case the segment is empty, distinguish between two cases:
  a If this is the first iteration of the steps (3) to (7), mark the triangle as invalid and go to (2).
  b If this is not the first iteration, use the last trajectory received at step (7) and go to (9).
- Use the process model to determine the covered area of the new trajectory segment. All positions that do not contribute any triangle to the planned area are removed from the new segment. In case the segment is empty, distinguish between two cases:
  a If this is the first iteration of the steps (3) to (7), mark the triangle as invalid and go to (2).
  b If this is not the first iteration, use the last trajectory received at step (7) and go to (9).

- One iteration of the local path optimization is performed to align the new segment with the inspected area or the border of the mesh.
- Repeat steps (3) to (7) until the local path optimization converges.
- Compute a collision-free connection from the current robot position to the start of the new segment.
- Move the robot along the new segment and inspect the newly planned area.
- If there are still valid triangles that are not yet inspected, go to (1). Otherwise the inspection is finished.

3.4 Real Applications

In this section, three applications of the proposed inspection framework are given. The framework has been initially applied in the context of automotive industry for visual inspection of cars engines directly on the production line. Following, the framework has been applied in the context of aerospace industry where X-Ray inspections are performed for inspecting the quality of the assembly of composite parts for detecting defects during their production. The remining application is then in the context of manufactury industry where two additional robot-sensor setups have been implemented to asses and demonstrate the real generality of the proposed approach: inspection of milled products with thermal imaging and inspection of metal forged parts using visual and laser inspection technologies. For each use case and scenario, qualitative and quantitative results will be given.

3.4.1 Inspections for the Automotive Industry

This section demonstrates the application of the proposed inspection framework in the context of automotive industry, more in particular, the experiment is focused on the visual inspection of car engines during their assembly on the production line for detecting the presence (or absence) of a component, the alignment (spatial orientation with respect to a notch or relative to another component), or compliance of the inspected part (the component mounted is the correct one).

A typical production line is depicted in Fig. 3.13. In such a production line, visual inspections are usually performed manually by a human operators, thus, the goal of the inspection

framework is to automatize this procedure using cooperative robots equipped with 3D sensors. In Figure 3.14 it is possible to see the real experimental setup developed at the facility of project partner *CRF*, it involves a Universal Robot UR10 cooperative robot equipped with a 3D depth camera and a reproduction of a small part of the real production line with the engine mounted on top of a movable conveyor. On the right of Figure 3.14 the reproduction of the same work-cell in the spirit off-line framework that has been used for generating the path for the inspection of the engine.

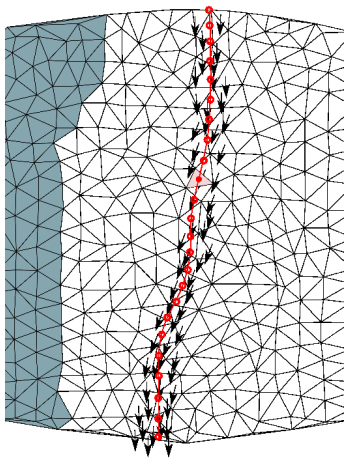
3.4.2 Inspections for the Aerospace Industry

In this experiment, the proposed inspection framework has been tested in a completely different scenario with respect the previous one so to demonstrate the high level of generality and wide range of applicability scenarios that can be reached with the proposed methodology. In particular, in this experiment a real X-Ray inspection facility has been involved. The robotic setup can be seen in Fig. 3.15 in which two Stäubli 200XL robots are used for carrying on the X-Ray sensor composed by a ray emitter tube and a sensor on the mirror robot that is able to detect the X-Rays. This setup has been thought for demonstrating the capability of the framework in achieving a continuous scanning operation with two robots moving synchronously while performing the inspection task. This experiment has been arranged to extensively test the Data Mapping algorithms proposed in Chapter 6.

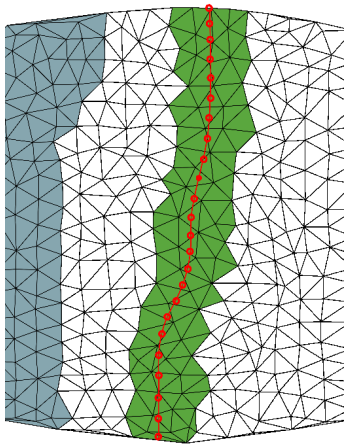
3.4.3 Inspections for the Manufactury Industry

This experiment has been one of the main achievements of the proposed inspection framework since it covers the majority of the possible use cases available in the industry. In particular, the experiment implements a 2D defect detection based on thermal camera imaging and a 3D inspection that involves a single line profilometer sensor. Pictures of the 2 use cases for the manufactury industry are given in Figure 3.16.

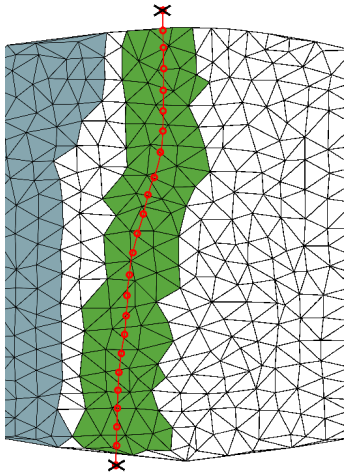
This use case has been developed to show the flexibility of the proposed inspection framework in modeling and then solving, in efficient way, complex defect detection problems for milled or forged parts, e.g. camshafts and other die-cast components.



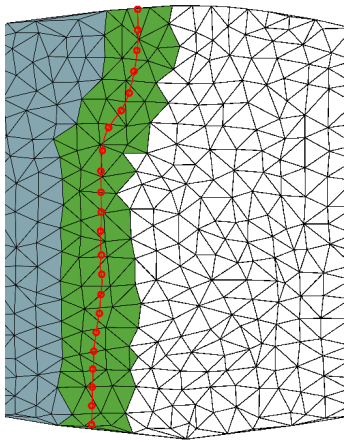
(a) Step (2) and (3) of the path extension: A triangle is chosen (light red) and its centre of mass (filled red circle) is the seed of a new stripe. The surface path of the new segments is generated by integration of the vector field (only plotted at the relevant triangles).



(b) At step (6) we compute the covered area (green) of the new stripe.



(c) A single iteration of the local path optimization is performed. The new stripe is pulled towards the border. At the next iteration of step (6) the coverage is evaluated again. Points at which no area is inspected are removed (in this example the first and last point of the segment).



(d) Step (8): The local path optimization converges and the new stripe is aligned with the inspected region.

Figure 3.12: Illustration of the path extension algorithm.

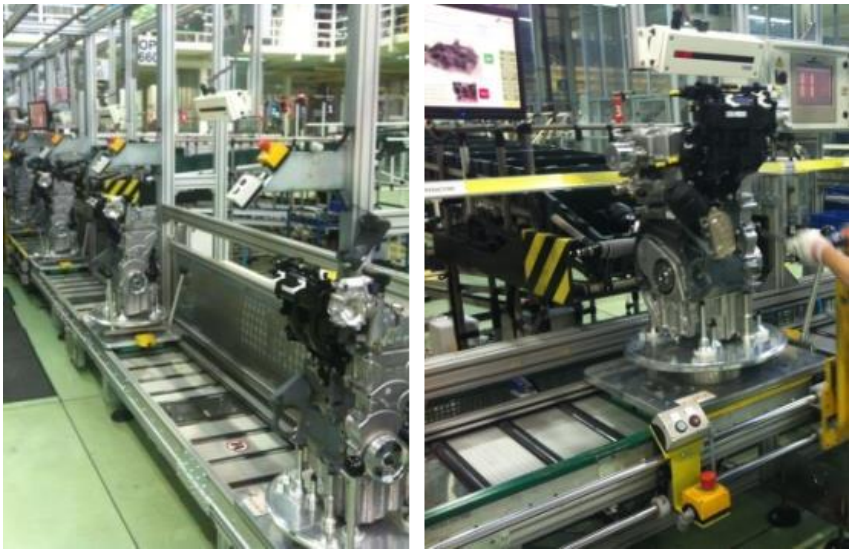


Figure 3.13: Typical engine production line examples. Courtesy of project partner *CRF*.

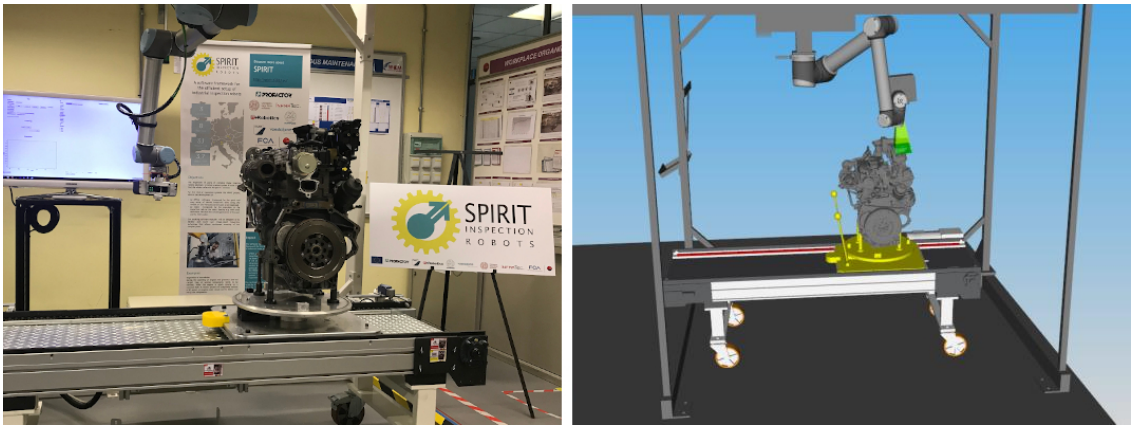


Figure 3.14: On the left the real experimental setup developed at project partner Centro Ricerche Fiat. On the right the simulated work-cell used in the offline framework.

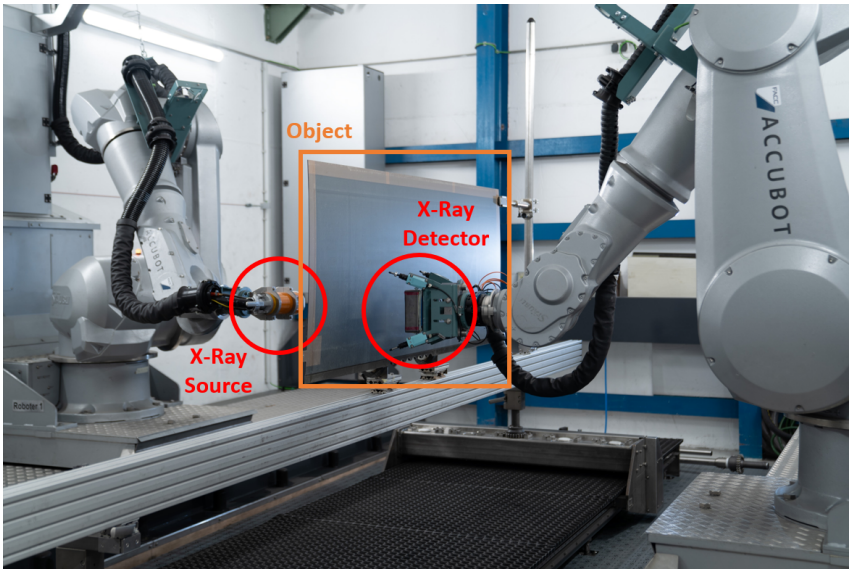


Figure 3.15: Layout of the X-Ray inspection workcell with detailed position of the X-ray source and detector and the part holders. Image courtesy of project partner FACC Gmbh.

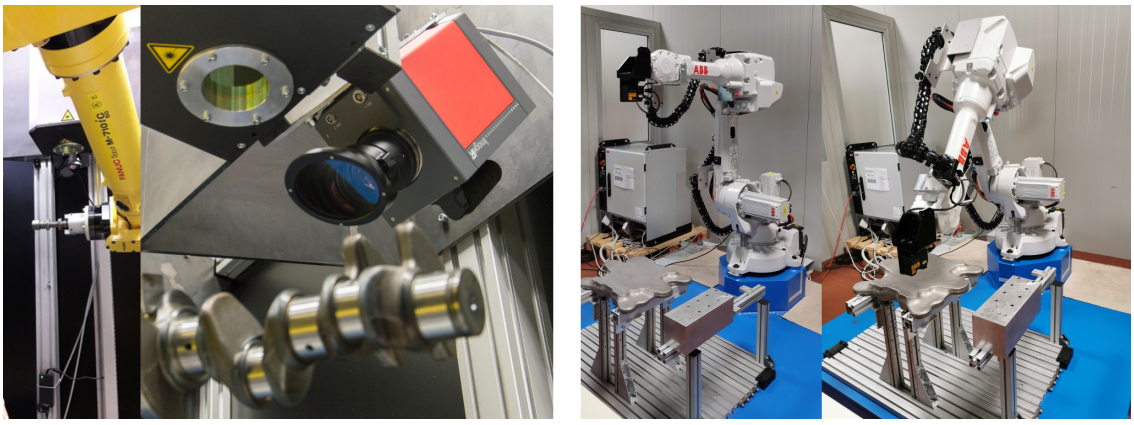


Figure 3.16: On the left, pictures of the workcell implemented to test the framework with thermal inspections. On the Right pictures of the workcell implemented to test the framework with laser profilometer inspections.

4

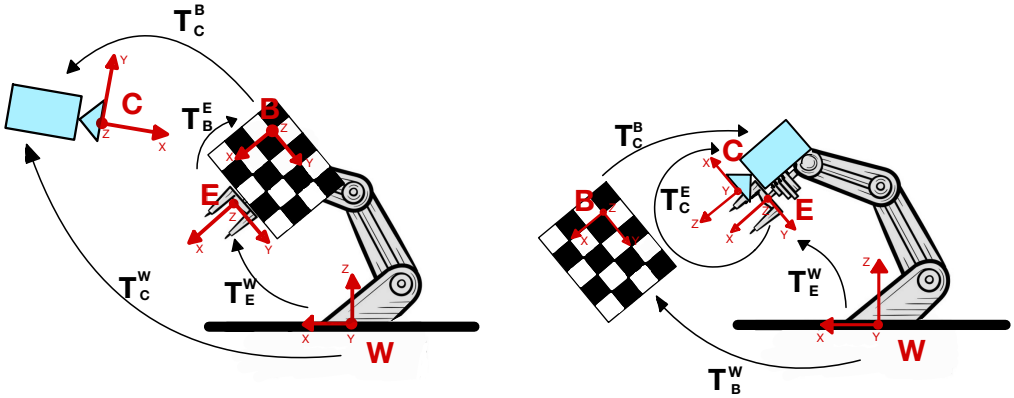
Local Hand-eye Calibration

This chapter describes the first approach of hand-eye calibration developed in this thesis work. As already mentioned in the introductory section of this thesis (see Chapter 1) this approach has been initially developed to overcome some of the difficulties encountered in applying already available hand-eye calibration techniques on the real application scenarios created to test the inspection framework developed in the context of the SPIRIT EU project that supported the first part of this thesis work.

In detail, all of the robotic workcells developed to test the inspection framework required a precise and accurate calibration of the sensor w.r.t. the robot reference frames. For this reason all the general approaches already proposed by [1, 2, 18] demonstrated to be not applicable to our scenarios, for this reason, a more local and iterative approach to solve the hand-eye calibration was necessary. This motivation brought to the development of the approach described in this chapter.

The chapter initially describes the analytical background of the proposed method in Sec. 4.1, then shows the results obtained in some experimental sessions, both in simulated conditions (see Sec. 4.2.1) and in real environments where the two main robot-sensor configurations have been tested (see Sec. 4.2.2 and Sec. 4.2.3 respectively).

The hand-eye calibration method described in this chapter has been published and presented in the *27th IEEE International Conference on Emerging Technologies for Factory Automation* held in 2022 in Stuttgart, Germany. See reference [40] for additional details on this publication.



(a) *Eye-on-base* setup illustration with a camera and a robot arm equipped by a checkerboard mounted on its end-effector.

(b) *Eye-in-hand* setup illustration with a robot arm equipped by a camera attached on its end-effector and a checkerboard in the robot's workspace.

Figure 4.1: Description of the transformations and reference frames for the two hand-eye configurations considered.

4.1 Methodology

As introduced before, the hand-eye calibration problem aims to determine the transformation between a robot end-effector and a sensor or between a robot base and the camera coordinate system. This problem is generally formalized as follows:

$$AX = ZB \tag{4.1}$$

where A is usually the camera-to-board transformation and B is the robot base-to-end-effector transformation given by robot kinematics, while X and Z are the two remaining unknown transformations (i.e. hand-eye and board-to-world transformations). In the literature, this problem generally relies on the PnP algorithm [41, 2] necessary to estimate the transformation between camera and checkerboard (i.e. A), which can be inaccurate and unreliable, since the pose of a calibrated camera is estimated considering a set of n 3D points in the world and their corresponding 2D projections in the image, which can be blurred, negatively affecting the entire hand-eye calibration.

Our proposed method does not rely on the PnP algorithm and solves the single-camera hand-eye calibration as a non-linear optimization, where the main aim is the minimization of the 2D distance between each reprojected 3D checkerboard's point $(u_x, u_y)_{ij}^{\text{proj}}$ on the image plane and the corresponding detected 2D corner $(u_x, u_y)_{ij}^{\text{det}}$, where i corresponds to the i^{th} corner of

the checkerboard and j denotes the j^{th} robot's pose achieved by the manipulator during the calibration.

Hence, the proposed hand-eye calibration method can be formalized with the optimization problem described in Eq. (4.2).

$$\underset{T_1,T_2}{\operatorname{argmin}} \sum_{j=0}^{M-1} \sum_{i=0}^{N-1} \left\| \begin{pmatrix} u_x \\ u_y \end{pmatrix}_{ij}^{\text{proj}} - \begin{pmatrix} u_x \\ u_y \end{pmatrix}_{ij}^{\text{det}} \right\|^2 \tag{4.2}$$

In particular, Eq. (4.2) represents the reprojection error of the N 3D corners of the checkerboard calculated as the difference between their respective projection (proj) and detection (det) on the M images captured at each achieved pose by the robot during the calibration process.

The minimization is computed according to the two transformations T_1 and T_2 that have to be estimated. These two transformation matrices variate according to the hand-eye configuration taken into account, and they are accurately described in Sections 4.1.1 and 4.1.2.

In Fig. 4.1 the robotic setups *eye-on-base* and *eye-in-hand* are shown. In both configurations, the main elements that contribute to the calibration process are the following reference frames:

- **W** represents the *robot base* reference frame, usually named *world*;
- **E** represents the *robot end-effector* reference frame;
- **B** represents the *checkerboard* reference frame;
- **C** represents the *camera* reference frame;

and the following transformations between the reference frames:

- T_C^W denotes the rototranslation between the world frame **W** and the camera frame **C**;
- T_E^W denotes the rototranslation between the world frame **W** and the robot end-effector frame **E**;
- T_B^E denotes the rototranslation between the robot end-effector frame **E** and the checkerboard frame **B**;
- T_C^B denotes the rototranslation between the checkerboard frame **B** and the camera frame **C**;

- T_C^E denotes the rototranslation between the end-effector frame **E** and the camera frame **C**;

- T_B^W denotes the rototranslation between the checkerboard frame **B** and the world frame **W**.

In the following sections, we will describe the proposed hand-eye calibration. A more detailed mathematical description of the reprojected 3D checkerboard's point $(u_x, u_y)_{ij}^{\text{proj}}$ will be given for the *eye-on-base* and *eye-in-hand* setups, in order to highlight how the optimization problem differs in the two hand-eye configurations. This is made on purpose to show the high flexibility and generality of our approach.

4.1.1 Eye-on-base

In the *eye-on-base* setup, the two rigid body transformations $T_1 = T_W^C$ and $T_2 = T_B^E$ are the two unknowns that must be optimized, the former in particular is the commonly known hand-eye transformation.

Some transformations in the optimization process are treated as constants and, therefore, they are not optimized during the entire calibration process, for example, T_E^W, which is retrieved by the kinematics of the robot arm, and T_C^B, which, as described in Sec. 2.2 is not taken into account in the optimization problem.

More in detail, given a set of N 3D control points defined as:

$$P_i^B = [X_i, Y_i, Z_i], \quad \text{with} \quad i = 0, \ldots, N-1 \quad , \tag{4.3}$$

which represent the 3D corners of the checkerboard expressed in the reference frame B, in the optimization problem described in Eq. (4.2), their reprojection onto the image plane is given by the following formula:

$$\begin{pmatrix} u_x \\ u_y \end{pmatrix}_{ij}^{\text{proj}} = K\,[I|0] * (T_W^C T_{Ej}^W T_B^E) * P_i^B \quad , \tag{4.4}$$

where K is the matrix of the intrinsic parameters of the camera in our experimental setup and T_{Ej}^W is the j^{th} pose of the robot end-effector expressed in the reference frame W.

4.1.2 Eye-in-hand

The formulation of the problem defined in Eq. (4.2), introduced in Sec. 4.1, applies exactly the same optimization to the *eye-in-hand* configuration. Here, the term *proj* relative to the reprojected corners must be specified accordingly by changing the role of some elements in the computation. In particular, in this case, this is the formulation that has to be applied:

$$\begin{pmatrix} u_x \\ u_y \end{pmatrix}_{ij}^{\text{proj}} = K\,[I|0] * (T_E^C T_W^{Ej} T_B^W) * P_i^B \tag{4.5}$$

where T_W^{Ej} is $(T_{Ej}^W)^{-1}$. Looking closely at Eq. (4.4) and Eq. (4.5), the way in which we adapt the optimization procedure described by the Eq. (4.2) from the *eye-on-base* setup to this scenario is the following:

- the hand-eye transformation is $T_1 = T_E^C$ which takes the place of T_W^C of Eq. 4.4;
- the position of the board in the reference frame W, that is, the transformation $T_2 = T_B^W$, replaces the term T_B^E in Eq. 4.4.

4.1.3 Non-linear Optimization

To solve the hand-eye calibration problem formulated in the previous sections, a non-linear iterative optimization has been used. In particular, we implemented our solution using the Ceres Solver [42]. More in detail, automatic differentiation has been used to define our cost function using the residual represented by Eq. (4.2). The Dense Schur solver has been used for solving the optimization problem driven by Huber loss function so that to reduce the influence of outliers on problem convergence.

4.2 Experiments and Evaluation

The proposed hand-eye calibration method has been tested in both simulated and real environments. In particular, in Sec. 4.2.1 we show the robustness and estimation accuracy of the proposed method by testing it under visual detection noise and initial guess pose noise. In Sec. 4.2.2 and Sec. 4.2.3 we will then demonstrate the effectiveness of the proposed hand-eye calibration approach in real camera-robot calibration tasks.

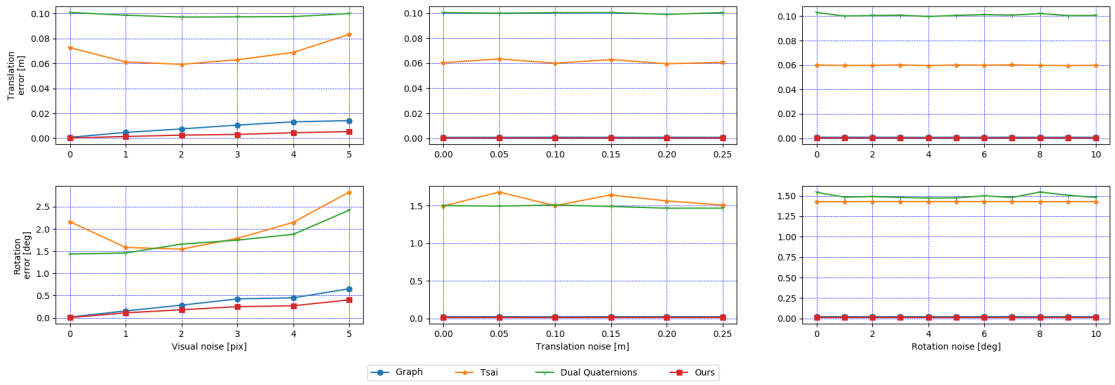


Figure 4.2: Result of the evaluation of the proposed hand-eye calibration method vs. Koide's method [1] (named *Graph* in the plots), Tsai's method [2], and the Daniilidis's method [3].

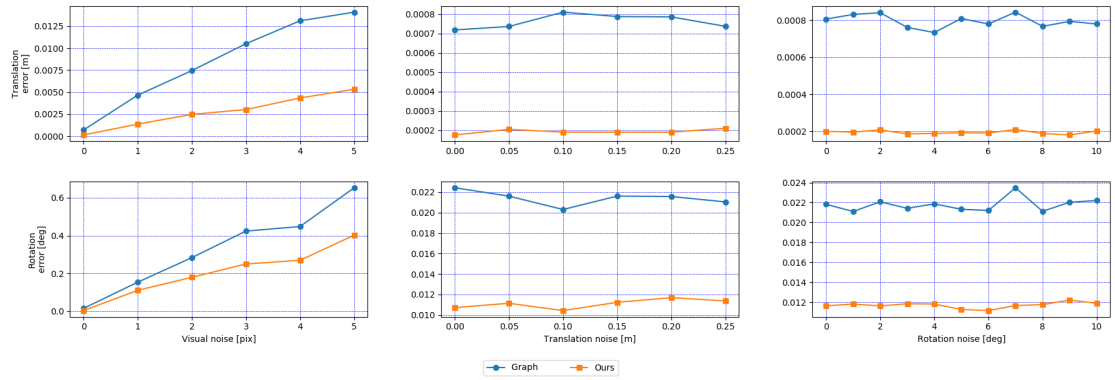


Figure 4.3: Focus on the result of the evaluation of the proposed hand-eye calibration method vs. Koide's method [1] (named *Graph* in the plots) only.

4.2.1 Experimental Evaluation

Using the same procedure described in [1], we tested the proposed approach by introducing, in simulated conditions, three different sources of noise: visual, translation, and rotation; this evaluation protocol aims to test the proposed calibration method under perturbed visual conditions (i.e., mimic errors or random noise in the detection of the calibration pattern) and aims to test the optimization accuracy and robustness when additional noise is introduced in translation and rotation of the initial guess of the hand-eye transformation.

For each simulation, we gradually introduced visual noise by perturbing the detected checkerboard corners in the range $(\Delta X, \Delta Y) \in [0, 5][\text{pixels}]$. Instead, for the initial guess perturbation, we tested the calibration by introducing perturbation transformations sampled from the following pose distribution: $\|t\| \in \mathcal{N}(\mu = 0, \sigma = 0.25)[m]$ and $\theta \in [0, 10][\text{deg}]$. In each

experimental run, nine camera stations have been generated at two heights with respect to the calibration pattern consisting of (7×5) points, to simulate a camera image acquisition at each station while looking at the calibration pattern. Thus, 18 images are generated for each simulation.

The proposed method, Koide's method [1], Tsai's method [2], and Daniilidis's method [3] are applied to estimate the hand-eye transformation. Using the described evaluation protocol, the simulation is performed 100 times for each noise setting.

In Fig. 4.2 the result of this evaluation study is depicted. Since the proposed method and Koide's [1] one outperform by an order of magnitude the other methods, in Fig. 4.3 the performances of the two methods have been reported for a better comparison. The proposed method shows good estimation results under both visual and initial guess noise, setting a new state-of-the-art in hand-eye calibration. In detail, the proposed method outperforms Tsai [2] and Daniilidis [3] by an order of magnitude, being even better with the highly competitive calibration result obtained by Koide [1].

4.2.2 Real Experiments – Eye-on-base

As previously mentioned, the proposed hand-eye calibration method has been tested in a real robotic cell. For the *eye-on-base* setup we have tested the proposed method against three methods from the literature, two of them are the same as in the experimental evaluation in Sec. 4.2.1, namely Tsai [2], and Daniilidis [3]. Due to the impossibility to adapt the official implementation of Koide [1], for this scenario we replaced this method with three additional methods, namely Andreff's [6], Seungwoong's [22] and the more recent hand-eye calibration method proposed in [43], where the authors use the 3D corner points of the calibration checkerboard to estimate a 3D affine transformation matrix between the 3D camera and the robot end effector. In particular to provide a faithful comparison with the method of [43], since they do not have published the relative calibration algorithm, we have precisely reproduced their setup with the same camera placed at the same distance from the robot arm.

The experimental setup is composed of a robotic workcell equipped by an *Emika Franka Panda*[1] robot manipulator and a *Microsoft Kinect v2* camera[2] placed at a distance $d = 1.10$ m from the robot base . A detailed picture of the proposed robotic workcell is given in Fig. 4.4.

The result in this test is given in terms of reprojection error, namely the difference, measured

[1] https://www.franka.de/

[2] https://developer.microsoft.com/it-it/windows/kinect/

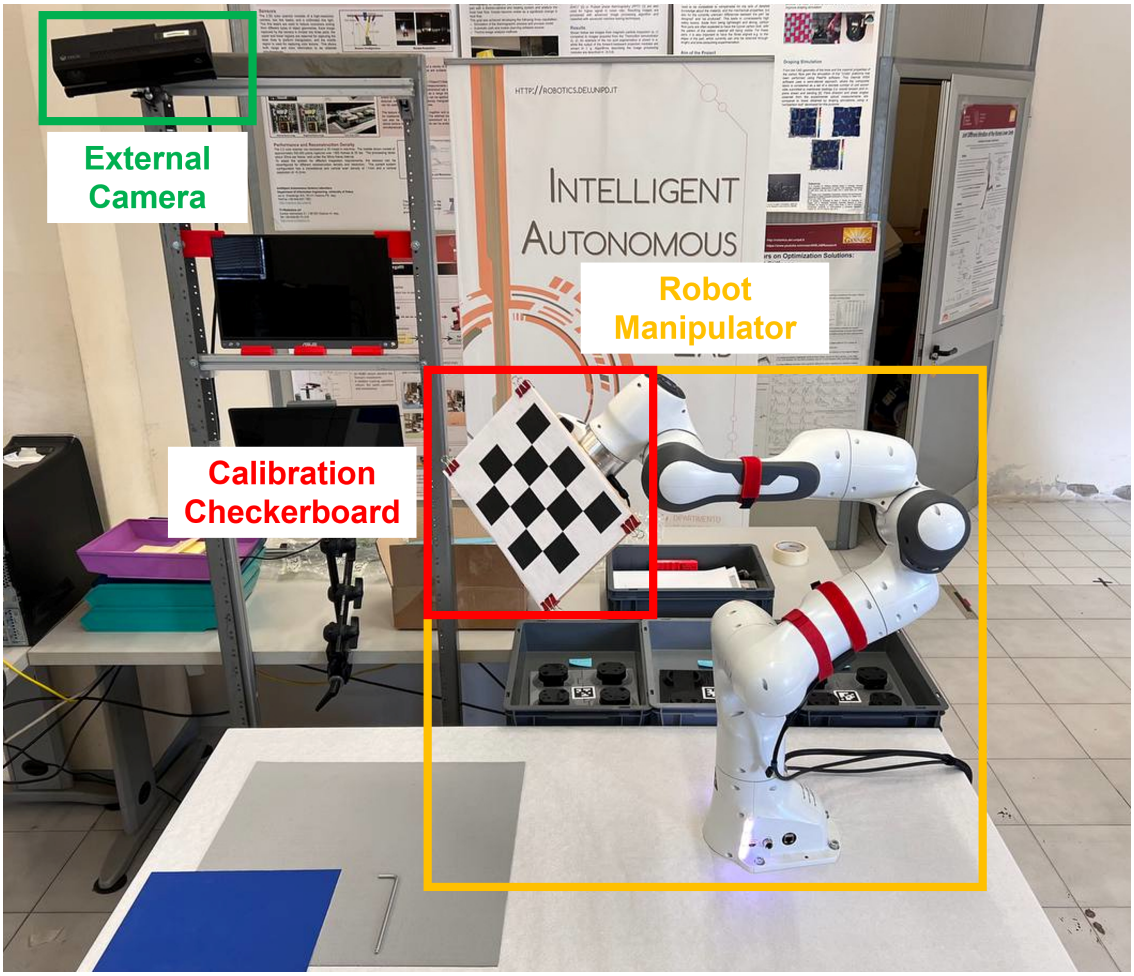


Figure 4.4: Picture of the real robotic setup used for the *eye-on-base* experiments.

in pixels, between the detected corners in the image plane w.r.t. the ones computed projecting their 3D coordinates onto the 2D image plane using the calibrated hand-eye transformation, as already explained in Sec. 4.1. In Tab. 4.1 the results of the experiments are given. In particular, the reprojection error obtained by each method and reported in the table is evaluated computing the average of the reprojection error obtained on all the stored images during the calibration. The proposed hand-eye calibration method overcomes all the other methods resulting to be more robust and accurate in the computation of the correct hand-eye transformation matrix.

Method	Reprojection Error [pix]
Tsai [2]	4.51
Daniilidis [3]	7.68
Andreff [6]	5.42
Seungwoong [22]	5.13
Mišeikis [43]	0.75
Ours	**0.27**

Table 4.1: Calibration results in *eye-on-base* setup.

Method	Reprojection Error [pix]
Tsai [2]	52.28
Daniilidis [3]	133.77
Koide's [1]	13.56
Ours	**6.47**

Table 4.2: Calibration results in *eye-in-hand* setup.

4.2.3 Real Experiments – Eye-in-hand

For the *eye-in-hand* test we performed a slightly different set of experiments. In particular a *ZED* camera[3] has been mounted to the end-effector of the robot - a picture of the proposed robotic setup is given in Fig. 4.5. It must be noticed that although the *ZED* is a depth camera, our method just exploits the RGB information. With this setup, the same methods proposed for the experimental evaluation in Sec. 4.2.1 have been tested; results are shown in Tab. 4.2. It can be seen that the proposed hand-eye calibration method outperforms the current state-of-the-art with a reduction of the reprojection error of around 50%. Our proposed approach outperforms other methods primarily due to our focused optimization on minimizing the re-projection error. Unlike most of the compared approaches that aim to optimize general and global constraints, such as homogeneous transformations, our approach prioritizes local and fine detail, resulting in superior performance in terms of numbers.

[3] https://www.stereolabs.com/zed/

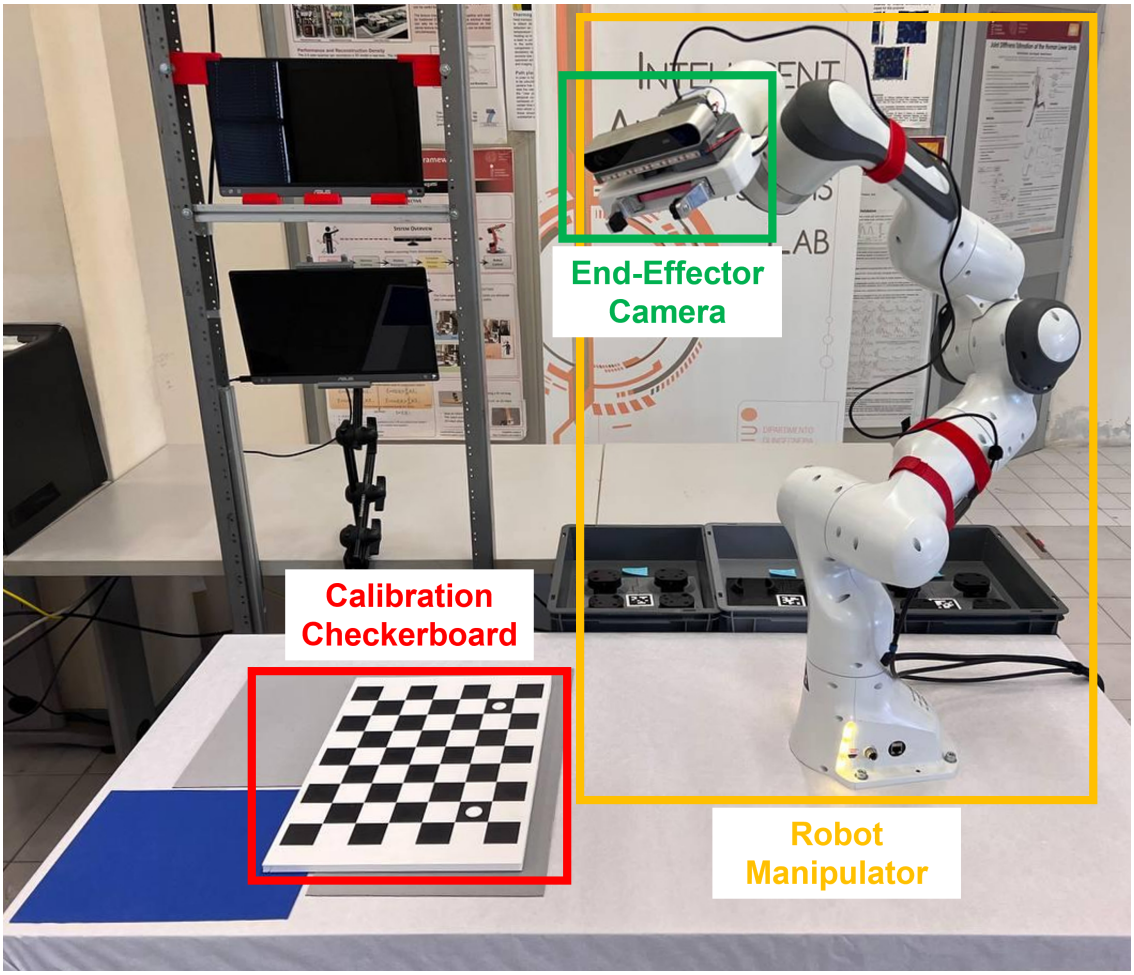


Figure 4.5: Picture of the real robotic setup used for the *eye-in-hand* experiments.

5

General Hand-eye Calibration

The approach for the hand-eye calibration described in this chapter is one of the key contributions of this thesis work. In particular, given the high complexity and heterogeneity of the industrial use cases addressed in this thesis activity, it was necessary to identify an appropriate approach for hand-eye calibration that could be easily adapted to solve all (or at least most) of the proposed robot-sensor setups.

This motivation brought us to the development of the hand-eye calibration described in this chapter. The approach overcomes the majority of the problems and limitations of all the hand-eye calibration techniques in literature, also for the one previously proposed in [40] and described in Chapter 4. This method relies on a *graph-based* formulation of the hand-eye calibration problem, and this is one of the key innovations that brought this method to overcome some of the major limitations since this high-level formulation is capable of solving different robot-sensor setups (e.g., *eye-on-base* and *eye-in-hand*) and, at the same time, it is easy to scale to multi-camera settings making it general enough to cover all the inspection workcells encountered during this thesis work.

The main contributions of the proposed general hand-eye calibration method are listed below:

(i) The proposed approach extends a recent state-of-the-art approach [1] by implementing a graph-based optimization solution to *eye-on-base* setups with one camera, not covered in the basic method;

(ii) Our method also covers multi-camera setups; thus, a general and unified graph-based optimization framework for hand-eye calibration is achieved;

(iii) An exhaustive evaluation of the proposed method is reported. The evaluation is primarily performed in simulated environments, where the robustness to visual and geometric noise can be tested. The proposed approach outperforms several popular calibration approaches, proving to be robust to high levels of noise in the calibration data. We also tested our method on a real setup, confirming the convincing results obtained in the simulations;

(iv) An open-source implementation of the proposed hand-eye calibration method is made publicly available at: `https://bitbucket.org/freelist/gm_handeye`

The chapter initially describes the analytical background of the proposed method in Sec. 5.1, then shows the results obtained in two different experimental sessions: in simulated conditions (see Sec. 5.2.1) and in a real environment where a multi-camera network has been calibrated w.r.t. an industrial robot manipulator (see Sec. 5.2.2).

The general hand-eye calibration method described in this chapter has been accepted as a regular paper in the *2023 IEEE International Conference on Robotics and Automation (ICRA)*. It will be presented in June 2023 after the first delivery of this thesis, for additional details on this publication please see the pre-print version already available in [44].

5.1 Methodology

This section presents a detailed overview of the proposed approach. As already mentioned, this work extends [1] in two main directions, explained hereafter: eye-on-base for single- and multi-camera setups. In this way, we are able to provide a complete framework that holds the necessary instruments to fulfil all the calibration needs that arise in robotic industrial applications.

5.1.1 Eye-on-Base Calibration for Single Camera

Here, we report the necessary notation for eye-on-base single camera setups.

- **W** : world reference frame, usually placed in the robot's base;
- **H** : hand reference frame, placed in the robot's end-effector;

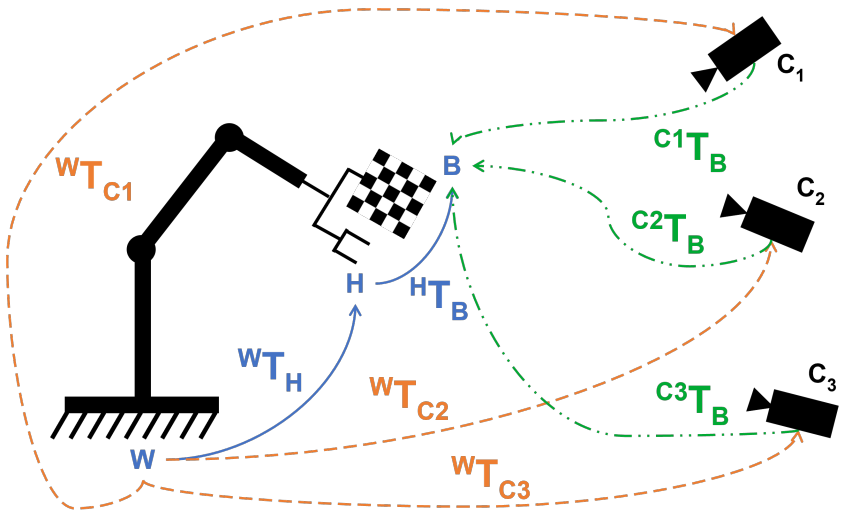


Figure 5.1: Multi-camera Eye-on-Base calibration setup. The goal of the proposed hand-eye calibration is to compute the position of each camera in the reference frame W, i.e., the transformations ${}^{W}T_{C1\ldots C3}$ in the figure.

- **B** : checkerboard reference frame;
- **C** : camera reference frame;

While the transformations are :

- ${}^{\mathrm{W}}\mathbf{T}_{\mathrm{H}}$: Isometry representing the pose of the end-effector in the world reference frame;
- ${}^{\mathrm{H}}\mathbf{T}_{\mathrm{B}}$: Isometry representing the pose of the checkerboard in the end-effector reference frame;
- ${}^{\mathrm{C}}\mathbf{T}_{\mathrm{B}}$: Isometry representing the pose of the checkerboard in the camera reference frame;
- ${}^{\mathrm{W}}\mathbf{T}_{\mathrm{C}}$: Isometry representing the pose of the camera in the world reference frame.

Fig. 5.2 shows the graph structure for the eye-on-base calibration. Using the notation from [1]: circles represent the state variables that need to be estimated, while hexagons are fixed parameters such as K and $P_{1..Z}$, that represent the camera matrix and the set of corners points of the checkerboard in their reference system, respectively. M represents the number of observations, i.e., images and robot poses, used for the calibration. The rectangles represent the different error terms. The error is generally defined as:

$$e_i(x) = h_i(x) \boxminus z_i, \tag{5.1}$$

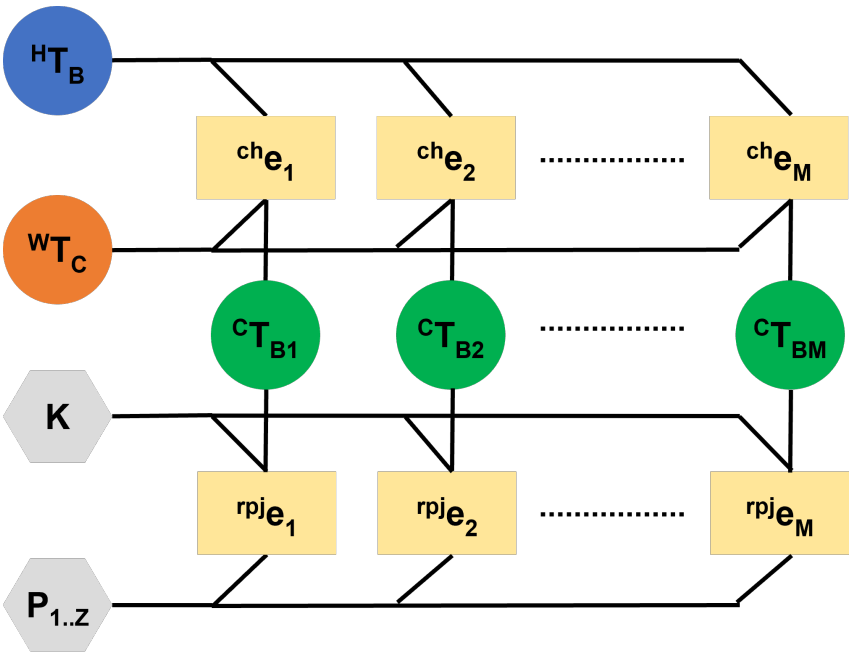


Figure 5.2: Graphical representation of the optimization problem for eye-on-base single camera setups. The representation contains circles, rectangles and hexagons that respectively represent the estimated variables, error functions, and fixed parameters. The error functions are defined following the equations 5.3 and 5.4.

where $h_i(x)$ is the measurement function and z_i is the i-th observation and $\boxminus$ is the difference operator for the manifold domain.

The first type of edge that is going to be presented is the one that enforces the equality :

$$ {}^{W}\mathbf{T}_{H} \cdot {}^{H}\mathbf{T}_{B} = {}^{W}\mathbf{T}_{C} \cdot {}^{C}\mathbf{T}_{B}, \tag{5.2} $$

that is one of the classic constraints for eye-on-base calibration [45]. And its corresponding error representation:

$$ {}^{ch}\mathbf{e}_i = t2v([{}^{H}\mathbf{T}_{W}]_i \cdot {}^{W}\mathbf{T}_{C} \cdot [{}^{C}\mathbf{T}_{B}]_i \cdot {}^{B}\mathbf{T}_{H}), \tag{5.3} $$

where the observation $\mathbf{z}_i = [{}^{W}\mathbf{T}_{H}]_i^{-1}$ is the i-th pose of the end-effector in the world. The remaining transformations are the ones that need to be estimated, while the $t2v(\cdot)$ is a function that converts the transformation to the corresponding minimal manifold representation in order to prevent optimization degradation due to over parametrization.

The second type of edge instead enforces the reprojection error constraint:

$$
{}^{rpj}\mathbf{e}_i = \begin{bmatrix} \pi([{}^{C}\mathbf{T}_B]_i \cdot {}^{B}\mathbf{P}_1) - [\mathbf{u}_1]_i \\ \vdots \\ \pi([{}^{C}\mathbf{T}_B]_i \cdot {}^{B}\mathbf{P}_j) - [\mathbf{u}_j]_i \\ \vdots \\ \pi([{}^{C}\mathbf{T}_B]_i \cdot {}^{B}\mathbf{P}_Z) - [\mathbf{u}_Z]_i \end{bmatrix}, \tag{5.4}
$$

where the observation $\mathbf{z}_i = [\mathbf{u}_1 \dots \mathbf{u}_j \dots \mathbf{u}_z]_i$ is a vector of tuples $\mathbf{u}_j = (u_x, u_y)_j$ that are the pixel coordinates of the j-th detected corner of the checkerboard at the i-th observation and π is the projection function of the camera.

The optimization aims to minimize the following function:

$$
\min \sum_{i=1}^{M} \left(\lambda \| {}^{rpj}\mathbf{e}_i \|^2 + \| {}^{cb}\mathbf{e}_i \|^2 \right), \tag{5.5}
$$

where λ is a weighting factor that accounts for the non homogeneity of the two error terms. The minimization is carried out by using the Levenberg-Marquardt [46] algorithm.

Our method proved to be robust also if provided with an identity transformation as initial guess, showing a rather large basin of convergence. This means that the system can be correctly calibrated even if we do not provide a good initial guess, which could be difficult to compute.

5.1.2 Eye-on-Base Calibration for Multi-Camera

If the robot is observed by multiple static cameras, one could repeat the above algorithm several times. On the other hand, running an algorithm that performs single-camera calibration N times, with N the number of cameras that are present in the workspace, is a sub-optimal solution because, in the optimization, the cross-observations between the different cameras are not factorized.

Given N cameras in the workspace, as shown in Fig. 5.1, there is the need to introduce a little more notation:

- ${}^{C_d}\mathbf{T}_{C_a}$: Isometry representing the pose of the a-th camera in the d-th camera reference frame;
- ${}^{C_a}\mathbf{T}_B$: Isometry representing the pose of the checkerboard in the a-th camera reference frame;

- ${}^{W}\mathbf{T}_{C_a}$: Isometry representing the pose of the a-th camera in the world reference frame.

The starting point for the multi-camera model is simply stacking the single-camera model N times as the number of the cameras. In Fig. 5.3 some of the vertices and edges have been grouped together in order to facilitate the graphical representation.

The error term ${}^{ch}\mathbf{e}_i^{1..N}$ enforces all the chained equalities of the system:

$$ {}^{ch}\mathbf{e}_i^{1..N} = \left[{}^{ch}\mathbf{e}_i^1 \dots {}^{ch}\mathbf{e}_i^a \dots {}^{ch}\mathbf{e}_i^N \right]^T, \tag{5.6} $$

where ${}^{ch}\mathbf{e}_i^a$ represents the chained equality for the a-th camera at the i-th time step.

The error term relative to the reprojection error evolves into:

$$ {}^{rpj}\mathbf{e}_i^{1..N} = \left[{}^{rpj}\mathbf{e}_i^1 \dots {}^{rpj}\mathbf{e}_i^a \dots {}^{rpj}\mathbf{e}_i^N \right]^T, \tag{5.7} $$

where ${}^{rpj}\mathbf{e}_i^a$ is the reprojection error term relative to the camera a-th at the i-th time step. Up to this point, we have simply stacked the single-camera graph N times one on top of the other, in fact the derivation of the new error terms was straightforward. The additional error term that is added to the graph model is the one that represents the cross-observation between cameras, from which information about their relative spatial relation can be extracted. Cross-observations occur when more cameras are able to see the checkerboard at the same time step i. In order to add it in an automatic manner to the optimization, it is needed to build a co-visibility matrix $\mathbf{X}_i$. At each time step i, each cell of this matrix is represented by $\mathbf{X}_i(a, d)$, where a and d are two different cameras (i.e., $a \neq d$). The cell contains a binary value that encodes whether or not the checkerboard is in the field of view of both cameras a and d at the i-th time step:

$$ \mathbf{X}_i(a, d) = \begin{cases} 0, & \text{if } a = d \text{ or no cross-observation} \\ 1, & \text{otherwise} \end{cases} \tag{5.8} $$

Thus, using the co-visibility matrix we are able to enforce the cross-observation constraints only where they are actually aiding the optimization.

To capture the spatial relation between cameras the transformation ${}^{C_d}\mathbf{T}_{C_a}$ is added as a state variable that needs to be estimated.

	TSAI [2]	PARK [4]	HORAUD [5]	ANDREFF [6]	DANIILIDIS [3]	SHAH [7]	LI [8]	GRAPH_MULTI
Camera 1	443.763	0.520446	0.432767	1.40595	196253	0.503051	0.603978	**0.337803**
Camera 2	0.416451	0.346938	0.35017	1.98045	0.538384	0.439182	0.717876	**0.284408**
Camera 3	148.89	26.2883	25.0635	165429	38.9004	97.8753	417.908	**4.4232**
Average	197.69	9.0519	8.61548	55144.2	65430.8	32.9392	139.743	**1.6818**

Table 5.1: Evaluation of the reprojection errors using the real setup data. All the values in the table are in pixel unit.

The new error term that models the cross-observation is defined as follows:

$$ {}^{cross}\mathbf{e}_{iad} = \begin{bmatrix} \pi_a({}^{C_a}\mathbf{T}_{C_d} \cdot [{}^{C_d}\mathbf{T}_B]_i \cdot \mathbf{P}_1^B) - [\mathbf{u}_1^a]_i \\ \vdots \\ \pi_a({}^{C_a}\mathbf{T}_{C_d} \cdot [{}^{C_d}\mathbf{T}_B]_i \cdot \mathbf{P}_j^B) - [\mathbf{u}_j^a]_i \\ \vdots \\ \pi_a({}^{C_a}\mathbf{T}_{C_d} \cdot [{}^{C_d}\mathbf{T}_B]_i \cdot \mathbf{P}_Z^B) - [\mathbf{u}_Z^a]_i \end{bmatrix}, \tag{5.9} $$

where the indices a and d refer to the cameras and $[\mathbf{u}_j^a]_i$ represents the j-th corner of the checkerboard detected by the camera a at the i-th time step.

This error accounts for the current estimate of the relative transformation between cameras. Checkerboard points are projected into the camera c_a using the camera c_d as starting point.

To simplify the notation all the error terms at time step i will be marginalized:

$$ {}^{cross}\mathbf{e}_i = \sum_{a=1}^{N}\sum_{d=1}^{N} \|{}^{cross}\mathbf{e}_{iad}\|^2 \cdot \mathbf{X}_i(a,d), \text{ with } a \neq d. \tag{5.10} $$

Fig. 5.3 shows the complete graphical model that includes these final constraints. The resulting function that needs to be minimized is the following:

$$ \min \sum_{i=1}^{M} \left(\lambda_1 \|{}^{rpj}\mathbf{e}_i\|^2 + \|{}^{ch}\mathbf{e}_i\|^2 + \lambda_2 {}^{cross}\mathbf{e}_i\right) \tag{5.11} $$

The effect that the cross-observations have on the optimization is to make it more robust against noise, and it will be effectively shown in the following section.

The proposed algorithm has been implemented using the g2o [47] framework. The weighting factors λ_1 and λ_2 have been experimentally tuned and set to the values $[10^{-6}, 10^{-3}]$, respectively, and kept fixed during all experiments.

5.2 Experiments and Evaluation

In Section 5.2.1 we show the robustness and estimation accuracy of the proposed method by testing it under visual and geometric perturbations. In particular, we both added noise to the detection of the checkerboard corners, thus simulating wrong detections of the calibration pattern (e.g., when using low-resolution cameras), and we also added noise to the robot poses $[{}^{H}\mathbf{T}_{W}]_i$ used for calibrating.

The proposed approach has been tested against the following methods (i.e., implementation from OpenCV[1]): [2, 4, 5, 6, 3, 7] and [8]. In the graphs and in the table we name them by reporting the first author of each one. The alternative methods have been tested by calibrating every single camera independently and then averaging the errors. This experimental methodology has been adopted because the alternative approaches are not directly usable for multi-camera calibration. The evaluation also reports an in-depth focus on the comparison of our multi-camera calibration method against the repetition of the proposed single-camera calibration applied to the individual cameras, this is done to demonstrate the robustness and improved accuracy of the multi-camera calibration framework.

5.2.1 Experiments with Synthetic Data

Using the same evaluation protocol proposed in [1, 40], our method has been tested, under simulated conditions, using three different sources of noise: visual, translation, and rotation. This evaluation protocol aims to test the robustness of the calibration under perturbed visual conditions (i.e., introducing noise in the detection of the corners of the calibration pattern), and it is also used to test the optimization accuracy and convergence capabilities when additional geometric noise is added to the robot poses.

We collected simulated data using the *Gazebo*[2] simulator, in which a simulated work cell has been implemented. The proposed work cell is composed of an industrial manipulator, mounting the calibration pattern, surrounded by five external cameras as in typical multi-camera eye-on-base setups. A set of 150 robot poses has been selected by randomly sampling the robot workspace to uniformly cover each camera field of view.

The results of this evaluation are reported in Fig. 5.4 and Fig. 5.5. In particular, in Fig. 5.4 the plots show the performance of all the calibration methods, while in Fig. 5.5 only the results

[1] http://docs.opencv.org/4.5.4/d9/d0c/group__calib3d.html
[2] https://gazebosim.org/home

of our single- and multi-camera graph-based calibration approaches are reported. The results show how our proposed method achieves good results in terms of translation and rotational errors even in strong perturbance conditions, demonstrating that it is more robust compared to other methods. The proposed eye-on-base single-camera calibration (i.e., GRAPH_SINGLE in Fig. 5.4 and Fig. 5.5) generally outperforms all the other tested approaches. Moreover, our multi-camera calibration hand-eye algorithm (i.e. GRAPH_MULTI in Fig. 5.4 and Fig. 5.5) is the best performing approach, in particular when high levels of noise are present in the calibration data.

5.2.2 Experiments on a Real Setup

The proposed algorithm has also been validated in a real robotic environment. The setup consisted of a Franka Emika Panda[3] robot manipulator and three Kinect2 cameras (we use only the RGB images) placed around the robot, as shown in Fig. 5.6 and Fig. 5.7. Unlike the simulated evaluation, where ground truth is available, in the real environment the absolute pose of each camera is not known a priori. Thus, to perform the test in the real setup, the reprojection error has been considered as a metric. In Tab. 5.1 the results of all approaches are reported.

Our approach is capable of correctly calibrating all cameras in the setup, also in challenging lighting condition, e.g., for cameras #1 and #3 (see Fig. 5.6). Most of the tested methods were unable to converge to a low reprojection error solution, while our method provided the lowest error by a large margin.

[3] https://www.franka.de/

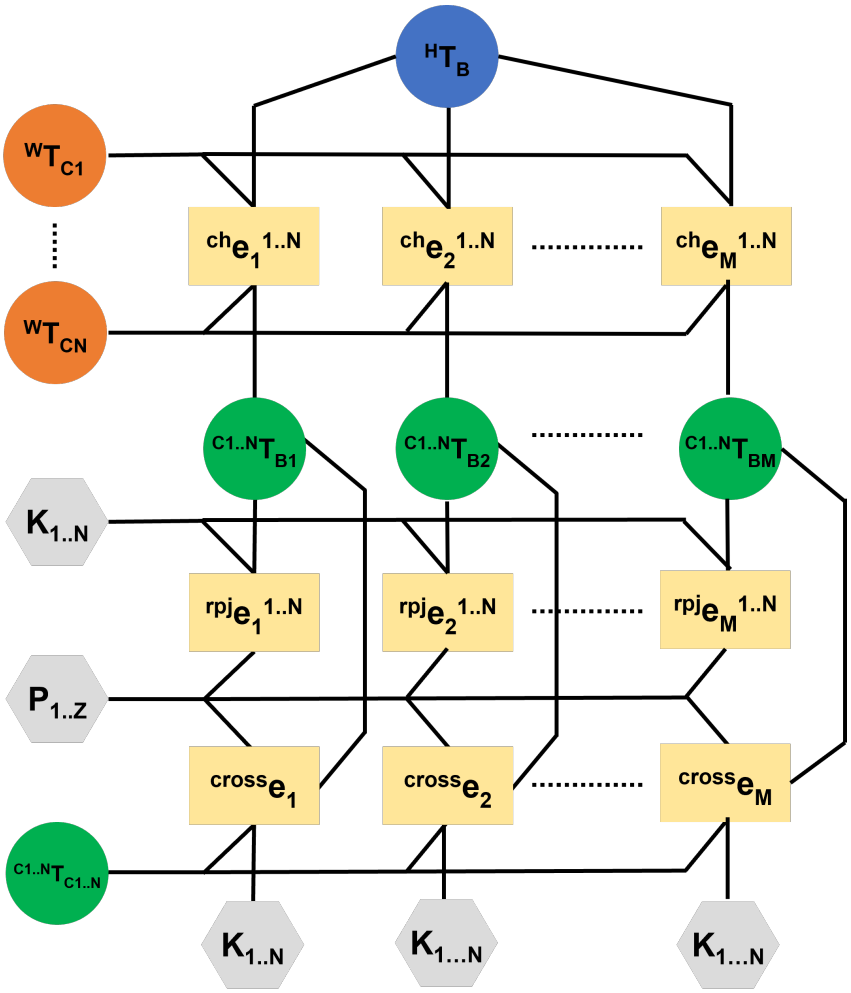


Figure 5.3: Evolution of the graph representation in multi cameras setups. The figure reports the error terms of Eq. (5.6-5.7) that are the stack of the errors in Eq. (5.3-5.4), respectively. Moreover, the graph includes an additional error defined in Eq. 5.10 which enforces constraints between the cameras with overlapping field of view.

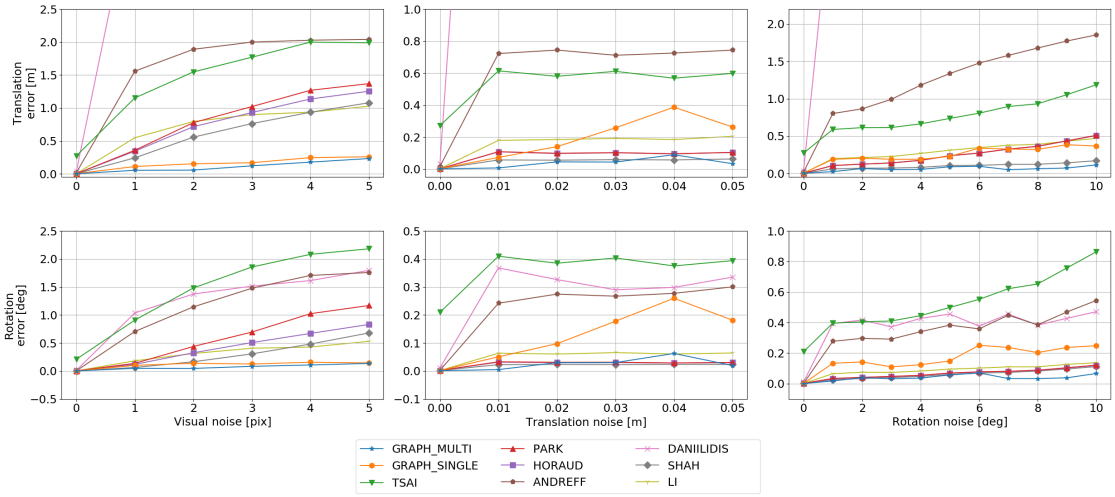


Figure 5.4: Result of the simulated evaluation. The proposed approach, both in *single-* and *multi-camera* settings has been tested against Tsai [2], Park [4], Horaud [5], Andreff [6], Daniilidis [3], Shah [7] and Li [8].

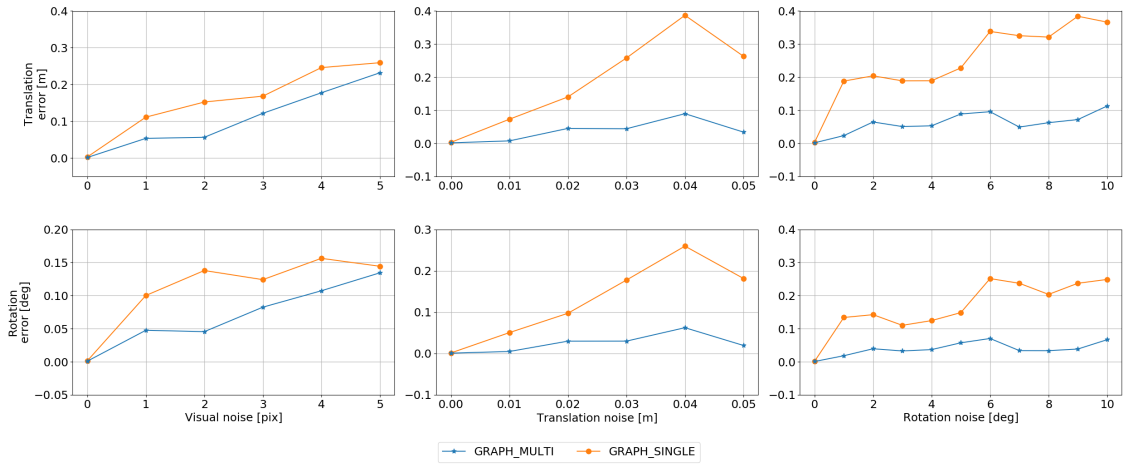


Figure 5.5: More in depth focus of the simulated evaluation regarding the proposed method only in *single-* and *multi-camera* settings.

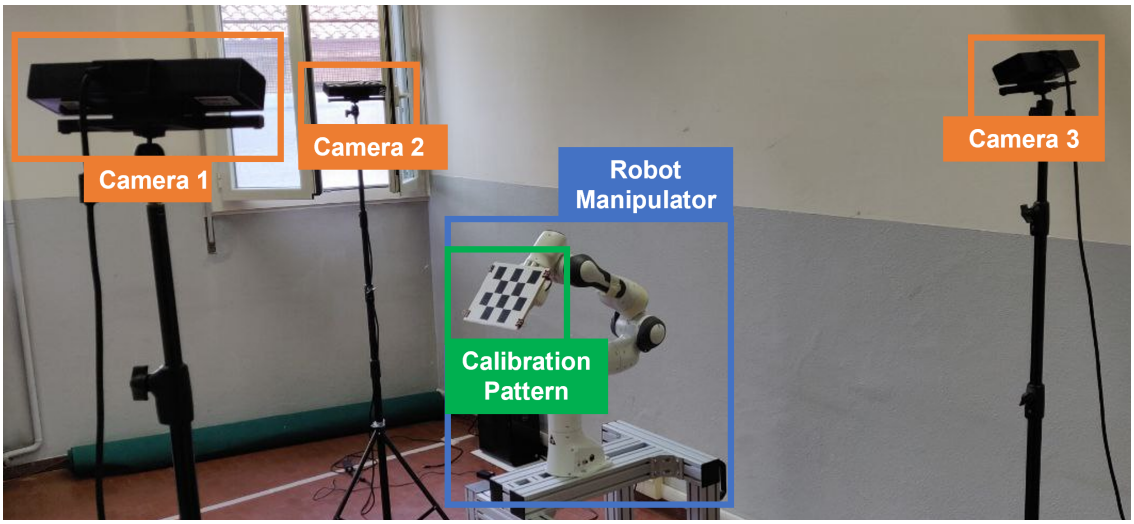


Figure 5.6: Front view of the real setup where the proposed calibration method has been tested.

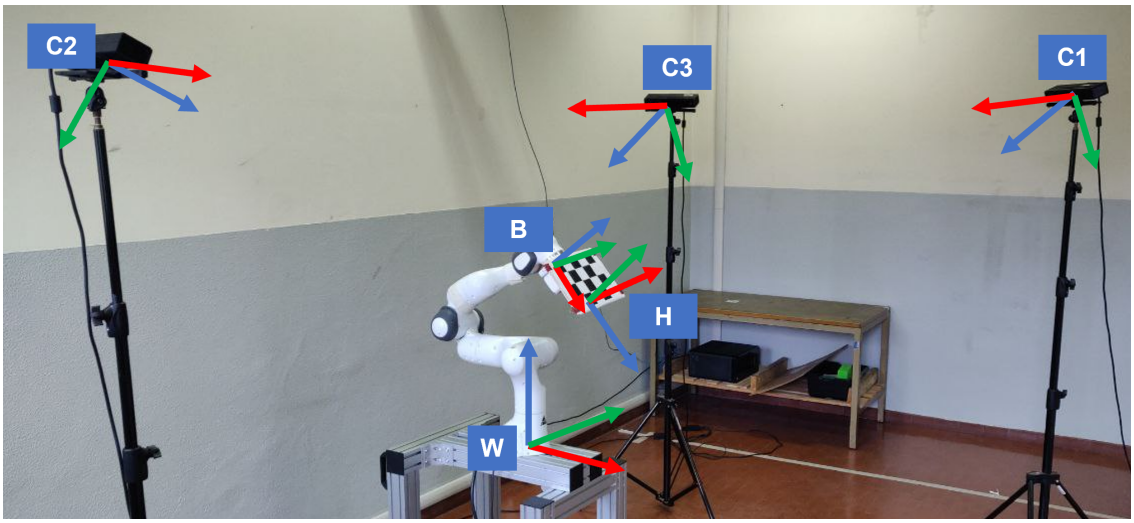


Figure 5.7: Rear view of the real setup. Main reference frames are depicted.

6
Data Mapping

As already anticipated in Chapter 1 one of the main contributions of this thesis work has been the study and development of a generalized approach for mapping sensor data over a 3D representation (i.e., CAD model) of the part to be inspected during an industrial inspection task. The reason why this method is described at the end of this thesis is that this software module, as also described in Chapter 3, requires an accurate calibration of the entire robotic workcell and the hand-eye calibration described in Chapter 4 and Chapter 5 is a key element which must necessarily precedes this module.

The Data Mapping (also named Data Backprojection) described in this chapter has been developed in the context of the SPIRIT project as part of the inspection framework described in Chapter 3. An intermediate application and development of the proposed approach has been published and presented at the *25th IEEE International Conference for Factory Automation (ETFA)* in 2020 and got the *Best Work-in-Progress Paper Award* for the excellent results obtained by the mapping approach in a aerospace industrial inspection task where a X-Ray sensor and two robot manipulators have been involved.

The chapter initially describes in Sec. 6.1 the Data Mapping approach and its formulation for the specific robot-sensor configurations. In Sec. 6.2 a sub-part of the whole Data Mapping pipeline is described in detail, it explains how the sequence of images (or data frames in general) are merged together for visualization purposes in a seamless way to avoid artifacts and strange mapping effects that may occur if no proper *stitching* is implemented. Finally, in Sec. 6.3 real applications of the proposed Data Mapping approach are presented.

6.1 Sensor Data Backprojection

Backprojecting sensor data from the 2D image (i.e., in case of camera inspections) to the 3D CAD model consists in computing a set of texture coordinates that map the visible surfaces of the object over the image plane. This procedure, in many inspection tasks, must fit within specific constraints; for example, the data mapping has to be performed in real-time and efficiently throughout the inspection process. Moreover, as already assessed for the computation of the inspection path in the offline framework (See Chapter 3 in Sec. 3.2) also data mapping must fit within some of the following constraints:

- **Process Model Parameters**
  - Camera distance from the object surface
  - Viewing angle (the angle between camera optical axis and object surface normal)
  - Masked image constraints (which part of the image need to be mapped)
- **Workcell Geometric Constraints**
  - Spacial transformations given by calibration of the part w. r. t. the robot base
  - Spacial transformations giveng by Hand-eye calibration

The above listed set of constraints is the key central element that makes the data backprojection approach shown in this chapter well integrated and fused within the proposed inline framework, process model parameters and calibration constraints are common information that both off-line and in-line framework share constantly.

6.1.1 Backprojection Algorithm

In Alg. 6.1 below we report the implementation of the backprojection algorithm implemented within the inspection framework described in Chapter 3. This algorithm computes the set of 2D pixel coordinates for each 3D vertex of the object mesh. In order to increase efficiency and reduce the computational cost for Alg. 6.1, the backprojection is done only for those triangles of the mesh that are visible from the given robot and sensor position and that fall within a given mask. This filtering procedure allows the algorithm to be executed in-line with the inspection process since it reduces a lot the number of total triangles of the 3D mesh that need

to be processed, moreover, it allows one to filter out part of the object that it is not required to be mapped.

The method proposed in Alg. 6.1 computes the texture coordinates instead of the color for each vertex of the 3D mesh. This aspect makes the whole process scalable since it does not require, like it happens when dealing with vertices, to have high-density 3D meshes in order to have seamless and continuous mapping of the object surface; instead it takes advantage of the interpolation done on triangle meshes at low level by the graphic engine when it applies the computed texture coordinates to the object even when the vertices of the 3D model are sparse.

Algorithm 6.1 Data Back-Projection algorithm.

Input: pose-referenced image set (I_S, O_S), CAD model *Obj*, *Mask* on the images
Output: the texture coordinates mapping *Map*

1: $Map \leftarrow \emptyset$
2: **for all** $(I_i, O_i) \in (I_S, O_S)$
3: $Pts_i = getVisiblePts(O_i, Obj)$
4: **for all** $P_j \in Pts_i$
5: **if** $P_j \notin Map$
6: $Im_i = applyMask(I_i, Mask)$
7: $(u, v)_j = unproj(P_j, Im_i)$
8: **if** $(u, v)_j \subseteq Im_i$
9: $Map = Map \cup [(u, v)_j, P_j]$
10: **else**
11: **continue**
12: **end if**
13: **else**
14: **continue**
15: **end if**
16: **end for**
17: **end for**

The result of Alg. 6.1 is a list of texture coordinates *Map* that is used as input to the graphical user interface developed within the inline software framework presented in Chapter 3.3. The Fig. 6.1 shows a reproduction of the developed software where the 3D model of a winglet (right part of the image) in the Aerospace Experiment (X-Ray setup, see Sec. 3.4.2) is mapped with image data coming from the X-Ray sensor (left part of the image).

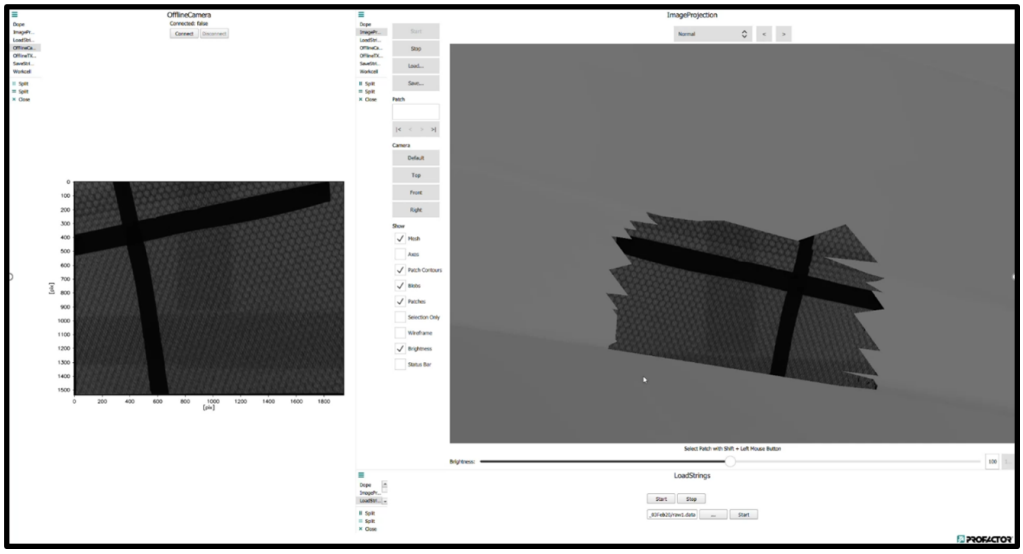
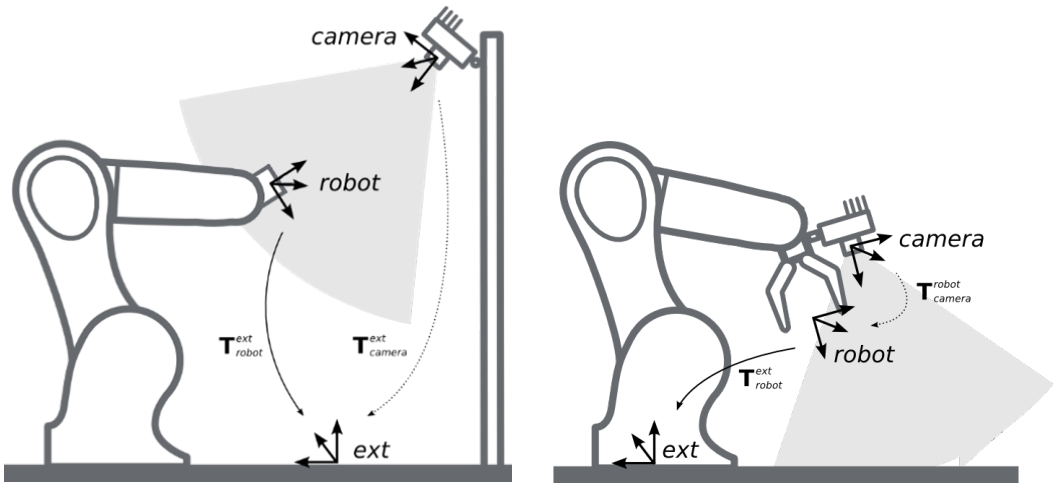


Figure 6.1: Example of the inline inspection framework GUI, visualization of the backprojection result.

Generalization Details

As already anticipated in previous sections, one of the key elements of this development has been the integration of the parameters of the process model and the details of the inspection process to make this block of the whole inspection framework well integrated and general enough to deal with the entire spectrum of applicability scenarios described in Sec. 3.4.

The process model parameters have been involved and integrated in one of the main functions of the pseudocode reported in Alg. 6.1, namely, the function in which we compute the $(u, v)_i$ texture coordinates for the i-th vertex. This part of the algorithm is the one that needs to be specified and tailored for the specific scenario, since it is the one that involves the mathematical operations for projecting 3D points onto 2D images and vice versa. That mathematical operation strongly depends on the projection model of the used sensor, e.g. RGB or monochrome cameras use the so-called Pinhole Camera Model, X-Ray use an Inverse Camera Model, etc. Moreover, depending on the robot sensor configuration that is involved, the projection operation can vary a bit; e.g. *eye-on-base* systems require an inverse input w. r. t. the *eye-in-hand* ones. All these aspects cannot be generalized and must be adapted to the specific setup. In the following we will go through the two robot sensor configurations declinating the proposed data mapping algorithm for each of the two setups.



(a) Schematics of the eye-on-base robot-sensor configuration.

(b) Schematics of the eye-in-hand robot-sensor configuration.

Figure 6.2: Schematics of the two robot-sensor configurations considered for the data mapping experiments.

Eye-on-base Configuration

The *eye-on-base* configuration is the one in which the sensor is mounted in a fixed position in the robotic workcell while the part to be inspected is mounted on the robot end-effector. In the context of this thesis project, this configuration has been used in the following experiments:

- **Inspections for the Manufactury Industry** (see Sec. 3.4.3). A specific thermal camera has been installed in a fixed position in the robotic workcell, its precise position has been calibrated using the hand-eye calibration procedure described in Chapter 4. The part to be inspected, namely a camshaft, contrary to the other experiments, has been mounted rigidly on the robot end-effector.

An example of the schematics for this robot-sensor setup is given in Fig. 6.2a. Using this configuration, a generic 3D point on the object part, expressed in the robot end-effector reference system (in Fig. 6.2a represented by *robot*) can be projected onto the 2D image plane of the camera using the following mathematical operation:

$$(u, v)_i = P_{cam} * T_{robot}^{camera} * T_{tcp}^{robot} * P_{tcp_i} \tag{6.1}$$

where:

- P_{cam} is the camera projection matrix.

- T_{tcp}^{robot} is the transformation from robot base reference frame to TCP reference frame (robot and ext respectively in Fig. 6.2a).
- T_{robot}^{camera} is the transformation from Camera reference frame to robot base reference frames (camera and ext respectively in Fig. 6.2a).
- $(u, v)_i$ and P_{tcp_i} are the (x, y) coordinates in the 2D image plane and the 3D point of the part expressed in robot end-effector reference frame respectively.

The Eq. (6.1) is one of the projection model implemented in the proposed backprojection algorithm and it actually reproduces the Pinhole Camera Projection where P_{cam} is taken as the Camera Intrinsics Matrix.

Eye-in-hand Configuration

The so-called *eye-in-hand* configuration is one of the common robot-sensor configurations available for inspection robotics cells. In the context of this thesis project, this configuration has been used in the following experiments:

- **Inspections for the Automotive Industry** (see Sec. 3.4.1). A 3D sensor used for inspecting the moving engines is mounted a UR10 robot hand.
- **Inspections for the Aerospace Industry** (see Sec. 3.4.2). The X-Ray sensor in this scenario is mounted on the hands of 2 Staubli robots.
- **Inspections for the Manufactury Industry** (see Sec. 3.4.3). In this experiment, the monochrome camera is mounted on a ABB robot hand while the part is fixed in the workcell.

An example of the schematics for this robot-sensor setup is given in Fig. 6.2b. In the configuration schematized in Fig. 6.2b the sensor is rigidly mounted on the robot hand, and the part is placed somewhere in the robot workspace. Using this configuration, a generic 3D point in the *robot_base* reference system (in Fig. 6.2b represented by *ext*) can be projected onto the 2D image plane of the camera using the following mathematical operation:

$$(u, v)_i = P_{cam} * T_{tcp}^{cam} * T_{robot}^{tcp} * P_{robot_i} \tag{6.2}$$

where:

- P_{cam} is the camera projection matrix.
- T_{tcp}^{cam} is the transformation from *TCP* reference frame to *Camera* reference frame (*robot* and *camera* respectively in Fig. 6.2b).
- T_{robot}^{tcp} is the transformation from *Robot* to *TCP* reference frames (*ext* and *robot* respectively in Fig. 6.2b).
- $(u, v)_i$ and P_{robot_i} are the (x, y) coordinates in the 2D image plane and the 3D point of the part expressed in *robot* reference frame respectively.

6.2 Image Stitching

Due to different influences when acquiring images position inaccuracies can occur that affect the result of the data mapping process. Such influences can arise from camera calibration, positioning of the workpiece or the robot position where the images are taken, etc. Therefore, the data mapping has to be extended by applying image stitching and calculation of correction vectors to compensate those incidents. Texture mapping is done on basis of triangle-meshes in the 3D world that are projected onto image coordinates, those coordinates are then distorted using template matching on the 2D image via small image parts and then those modified coordinates are used to re-project the image patches onto the 3D mesh.

6.2.1 Appending a new image

Every time a new image is added, a template matching is done to determine the correction of all visible key points in reference to already captured images. As template matching is prone to similar appearing key points on the work piece surface, the template matching is constrained to a limited part of the incoming image. The constraints for template matching and the template matching is explained in this section, as well as the calculation of the corrections for the visible key points and the resulting modification of the key point coordinates. These steps occur every time when an image is added.

Key points in this context are the points of the 3D mesh that represents the objects surface that shall be image captured and visualized (see Fig. 6.3).

At first the centre of mass of all visible key points is calculated whereby the masses of all key points are the same. Through the centre of mass of the key points of the current image the distance to the centre of mass of already captured images can be determined. This distance is

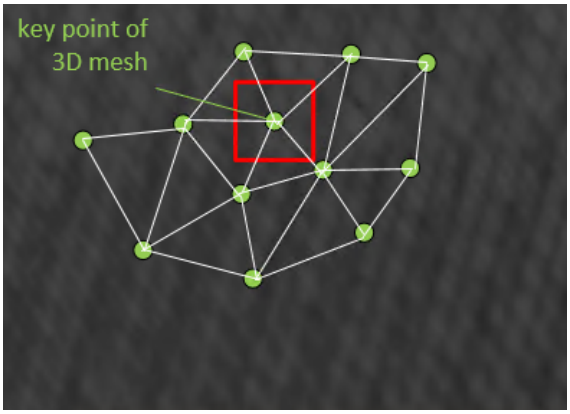


Figure 6.3: Representation of keypoints for the image stitching. A Key Point is a point of the 3D mesh that represents the objects surface.

then used to find a limited number of images whose positions were closest to the positioning of the incoming image.

For each visible key point in the incoming image a lookup is done in order to find out if one of those nearest images had the same key point. If this is not the case, the key point is marked as outlier. If one or more older images had the same key point, the best suited image is chosen based on the *quality* which is the variance of correction vectors in the 1-Ring key point neighbourhood of the current examined key point. The calculation of this *quality* score is described in detail in the Sec. 6.2.2.

The already known key points of this resulting best old image are then used for template matching on the current incoming image. Template matching is described in detail in Sec. 6.3. When the key point is detected in the current incoming image, the original coordinates are used to calculate the correction vectors in x and y direction for each key point inside the incoming image. On all those key point correction vectors inclusive the outliers that were not useful within the previous processing a kriging interpolation and convergence treatment is executed. This is done using the Gaussian Regression Process[1]. By applying this step the measurement noise can be used additionally to smooth outliers by the calculated variance of the surrounding key point correction vectors and also to smooth the calculated correction vectors. With the resulting correction vectors the key point coordinates can then be modified for better reprojection of the image onto the 3D mesh. This modification is not only applied to the key point coordinates of the incoming image but also the already modified key point coordinates of older images using corresponding weights. For example there were 9 old images and 1 incoming image, then

[1] http://www.gaussianprocess.org/

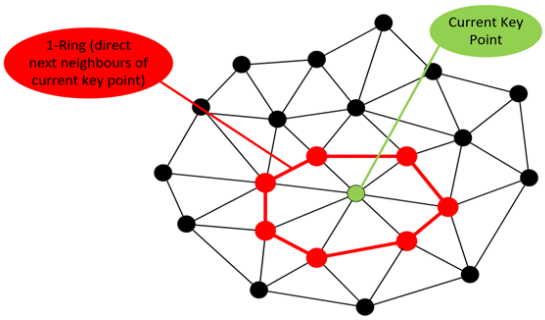


Figure 6.4: Example of the ring structure used for image stitching and computation of the correction vectors of the texture to be mapped during the data mapping process.

90% of the modification of key point coordinates will be done on the incoming image and the remaining 10% modification is done on the 9 older images. This weighting is important in case if the older images had wrong key point coordinates themselves compared to the key point coordinates of the incoming image. After modification a calculation of the new 1-Ring quality is done for each key point in the 3D mesh for further use when the next new image is captured.

6.2.2 Calculating the 1-ring quality

For determination of the best suited old image to calculate the correction vectors of key points in the new image, the quality of correction vectors in the old images is used. This quality is calculated from the variance of the already known correction vectors in the old images. See Fig. 6.4 for definition of a 1-Ring.

Each of these key points inside the old image has already calculated correction vectors. In order to find the best suited old image from which the correction of the key points inside the new image is derived, the variance of the correction vectors of the old image is used. If the variance of all the correction vectors in the 1-Ring inside the old image is low the correction vector for the key point in this old image is considered to be trustworthy and therefore is considered to be of good quality for calculation of correction. If the correction vectors in the 1-Ring inside the old image is high the correction vector for the key point in this old image is considered to be not trustworthy, as this means that the surrounding image piece around the key point could be too much shifted or rotated in respect to the real world and therefore is not suitable for further application of correction. See Fig. 6.5 for a visualization of good or bad quality.

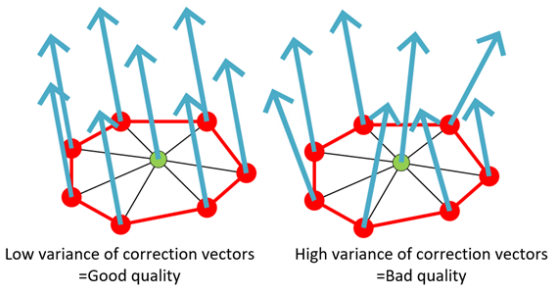


Figure 6.5: Visualization of the *quality* score computed for image stitching.

6.2.3 Template matching and Correction vectors computation

Template matching is not done over the whole incoming image but instead only done in an area that is restricted to the maximum translation distance which was defined in the settings. This maximum translation distance is necessary to exclude possible another template matches inside the incoming image. A visual example of this template matching process can be seen in Fig. 6.6.

In order to apply the current corrections of key point coordinates when reprojecting the image onto the 3D mesh an interface has been implemented. This returns the current corrections that were calculated each time a new image had been added. For runtime efficiency this request is only done on the last few images except if requested otherwise for example every e.g. tenth image the corrections of the key point coordinates is requested for all previous images instead of only the last ones.

6.3 Experimental Results

In this section, some qualitative results of the data mapping algorithm be shown. More in detail, it will be shown how the backprojection method developed within this thesis work has been successfully applied at the different experiments and use cases already described in Chapter 3. As already mentioned, the proposed inspection framework requires specific configurations to be adapted to each experiment, and the Data Mapping algorithm follows the same approach being an integrated part of it, e.g., the definition of the type of robot sensor configuration (i.e., *eye-on-base* or *eye-in-hand*) is one of the main parameters to be set in this software block. The next sections describe and present the data mapping results obtained in the two provided configurations.

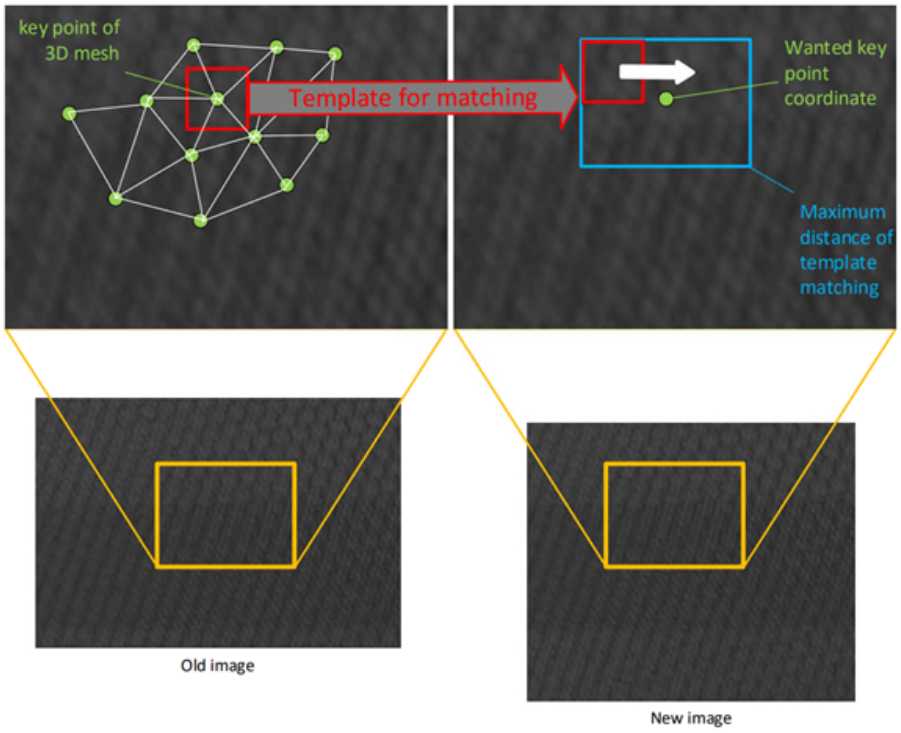


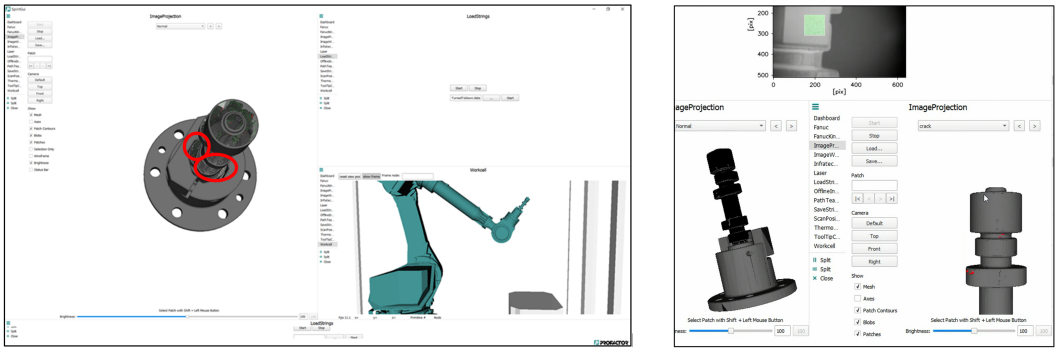
Figure 6.6: Template matching and maximum distance of template matching.

6.3.1 Mapping with Eye-on-base Setups

In this experiment we implemented the mathematical data backprojection principles described in Sec. 6.1.1. In this case the backprojection algorithm has been successfully applied and qualitative initial results can be seen in Fig. 6.7.

From Fig. 6.7a it is clearly visible how the initial implementation of the backprojection algorithm did not consider the possibility to exclude from the mapping the so called self-occluded surfaces. In particular, we consider as self-occluded face each triangle of the CAD mesh that is not visible by the camera because occluded by another triangle mesh closer to the camera itself. To avoid that occluded surfaces are mapped with wrong textures, the backprojection algorithm has been further updated by including all the parameters coming from the process model, more in detail, the final version of the backprojection algorithm takes into account also the following set of parameters retrieved by the common process model shared between the proposed offline and inline inspection frameworks:

- *Distance_range*: defines the distance that must exists between the sensor and the object



(a) (b)

Figure 6.7: Backprojection results in the *eye-on-base* experiment with thermal imaging inspection. The pictures show a visualization of the proposed software GUI within the inspection framework. Highlighted in red we have occluded surfaces (a) and detected defects (b).

surface. All the triangle meshes that are visible by the camera, but that fall outside this range, are not considered valid in the mapping procedure.

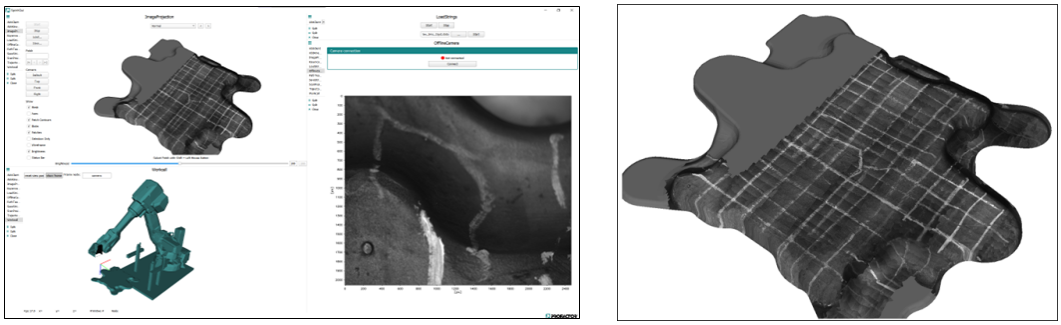
- *Angular_range*: defines the range of *camera principal axis - object surface normal* angles that are considered valid during the inspection procedure. As like as in the coverage planning described in Sec. 3.2.2, the same values are considered here, only triangle meshes whose normal vector does not exceed the angular threshold are considered valid for the mapping.

This implementation avoids the backprojection of textures onto self-occluded surfaces. Results can be seen in Fig. 6.7b, where the mapping is applied only to image regions where actual defects are detected by an external defect detection algorithm.

6.3.2 Mapping with Eye-in-hand Setups

In this experiment we implemented the mathematical data backprojection principles described in Sec. 6.1.1. In this case the backprojection algorithm has been successfully applied and qualitative initial results can be seen in Fig. 6.8.

The tests have been performed with forged parts provided by the project partner BSTG. In Fig. 6.8b a more detailed view of the backprojection results is shown, an artificial hand-made texture has been applied on the real object surface, the obtained results show how the backprojection is accurate enough to correctly reproduce the texture of the object without applying any particular stitching or data fusion method.



(a) (b)

Figure 6.8: Backprojection results in the *eye-in-hand* experiment with standard visual inspection with an industrial camera. The pictures show a visualization of the proposed software GUI within the inspection framework (a) and a detailed view of the obtained mapping results on the final textured 3D mesh (b).

7 Conclusion

This thesis addressed the problem of making industrial inspections with robot manipulators more robust and efficient by proposing general software modules to solve the problems of hand-eye calibration and data mapping. In particular, knowing the exact position of the sensor that performs the inspection task with respect to the robot reference frame (e.g., either the base or the end-effector frame) is a key fundamental element of a correct inspection. Accurate calibration obviates the necessity of intricate and resource-intensive recovery procedures for erroneous inspection path computations. Additionally, it ameliorates, from the outset, the impact of incorrect data mapping associations whereby certain areas of the object may not be adequately inspected due to inaccurate re-projection caused by system miscalibration.

To this end, in this thesis, we initially illustrate in a general inspection framework that makes use of hand-eye calibration and data mapping software blocks to address multiple inspection technologies and different robot sensor setups. Hand-eye calibration has been addressed by presenting a method for local iterative optimization and a more comprehensive approach using global graph-based optimization. In particular, the latter hand-eye calibration technique sets a new state-of-the-art by accurately solving multiple calibration problems; moreover, we were also able to provide a more general formulation of the same problem in the context of multi-camera robotic workcells, everything considered in a unified graph optimization framework.

The graph-based hand-eye calibration method presents a valuable source of inspiration for future research endeavors that require a broad approach to solving pose refinement and calibration problems. This approach draws inspiration from pose-graph optimization techniques

commonly employed in Simultaneous Localization and Mapping (SLAM) problems, which entail fusing global and local features within a unified optimization framework to achieve precise and accurate outcomes. Leveraging the inherent generality exhibited by the graph structure, our approach encompasses both global, such as geometric constraints among homogeneous transformations, and local, such as reprojection error-based image constraints, optimization to attain accurate results as demonstrated in both simulated and real-world experiments.

The graph proposed for calibration also offers flexibility for addressing other calibration problems, including but not limited to robot-to-robot calibration within multi-camera networks. By extending the graph-based framework, it becomes possible to accommodate the complexities and requirements of various calibration scenarios, enabling researchers to explore and solve diverse calibration challenges in a systematic and effective manner. This adaptability and potential for further extensions highlight the significance of the proposed approach and its applicability beyond hand-eye calibration, paving the way for advancements in the field of pose refinement and calibration research.

This approach has been published at the *International Conference on Robotics and Automation* and a public open-source implementation has been made available.

If a precise and accurate calibration of the robotic setup is achieved, the final building block of the general inspection framework, the *Data Mapping*, can be performed more easily. In this thesis, we have proposed a Data Mapping method that takes advantage of a precise and accurate hand-eye calibration result to address the problem of Data Mapping for multi-purpose inspection robots. This approach has been applied in multiple inspection setups, ranging from automotive to aerospace and manufactury industry.

References

[1] K. Koide and E. Menegatti, "General hand–eye calibration based on reprojection error minimization," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1021–1028, 2019.

[2] R. Tsai and R. Lenz, "A new technique for fully autonomous and efficient 3d robotics hand/eye calibration," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 345–358, 1989.

[3] K. Daniilidis and E. Bayro-Corrochano, "The dual quaternion approach to hand-eye calibration," in *Proceedings of 13th International Conference on Pattern Recognition*, vol. 1, 1996, pp. 318–322 vol.1.

[4] F. Park and B. Martin, "Robot sensor calibration: solving ax=xb on the euclidean group," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 5, pp. 717–721, 1994.

[5] R. Horaud and F. Dornaika, "Hand-eye calibration," *The International Journal of Robotics Research*, vol. 14, no. 3, pp. 195–210, 1995.

[6] N. Andreff, R. Horaud, and B. Espiau, "On-line hand-eye calibration," in *Second International Conference on 3-D Digital Imaging and Modeling (Cat. No.PR00062)*, 1999, pp. 430–436.

[7] M. Shah, "Solving the robot-world hand-eye calibration problem using the kronecker product," *Journal of Mechanisms and Robotics*, vol. 5, no. 3, p. 031007, 2013.

[8] A. Li, L. Wang, and D. Wu, "Simultaneous robot-world and hand-eye calibration using dual-quaternions and kronecker product," *International Journal of the Physical Sciences*, vol. 5, no. 10, pp. 1530–1536, 2010.

[9] D. Fusaro, E. Olivastri, D. Evangelista, M. Imperoli, E. Menegatti, and A. Pretto, "Pushing the limits of learning-based traversability analysis for autonomous driving on cpu," in *Intelligent Autonomous Systems 17*, 2023, pp. 529–545.

[10] E. N. Malamas, E. G. Petrakis, M. Zervakis, L. Petit, and J.-D. Legat, "A survey on industrial vision systems, applications and tools," *Image and Vision Computing*, vol. 21, no. 2, pp. 171 – 188, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S026288560200152X

[11] P. Kopardekar, A. Mital, and S. Anand, "Manual, hybrid and automated inspection literature and current research," in *Integrated Manufacturing Systems*, 1993.

[12] J. C. Noordam, G. W. Otten, T. J. M. Timmermans, and B. H. van Zwol, "High-speed potato grading and quality inspection based on a color vision system," in *Machine Vision Applications in Industrial Inspection VIII*, K. W. T. Jr. and J. C. Stover, Eds., vol. 3966, International Society for Optics and Photonics. SPIE, 2000, pp. 206 – 217. [Online]. Available: https://doi.org/10.1117/12.380075

[13] M. Abdullah, S. Aziz, and A. Mphamed, "Quality inspection of bakery products using a color machine vision system," *Journal of Food Quality - J FOOD QUAL*, vol. 23, pp. 39–50, 03 2000.

[14] E. Weigl, S. Zambal, M. Stöger, and C. Eitzinger, "Photometric stereo sensor for robot-assisted industrial quality inspection of coated composite material surfaces," in *Twelfth International Conference on Quality Control by Artificial Vision 2015*, F. Meriaudeau and O. Aubreton, Eds., vol. 9534, International Society for Optics and Photonics. SPIE, 2015, pp. 367 – 374. [Online]. Available: https://doi.org/10.1117/12.2182750

[15] G. Paul, S. Webb, D. Liu, and G. Dissanayake, "Autonomous robot manipulator-based exploration and mapping system for bridge maintenance," *Robotics and Autonomous Systems*, vol. 59, no. 7, pp. 543 – 554, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889011000704

[16] M. Imperoli, C. Potena, D. Nardi, G. Grisetti, and A. Pretto, "An effective multi-cue positioning system for agricultural robotics," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3685–3692, 2018.

[17] Y. Shiu and S. Ahmad, "Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form ax=xb," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 1, pp. 16–29, 1989.

[18] R. Tsai and R. Lenz, "Real time versatile robotics hand/eye calibration using 3d machine vision," in *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, 1988, pp. 554–561 vol.1.

[19] E. Marchand, F. Spindler, and F. Chaumette, "Visp for visual servoing: a generic software platform with a wide class of robot control skills," *IEEE Robotics Automation Magazine*, vol. 12, no. 4, pp. 40–52, 2005.

[20] J. C. K. Chou and M. Kamel, "Finding the position and orientation of a sensor on a robot manipulator using quaternions," *The International Journal of Robotics Research*, vol. 10, no. 3, pp. 240–254, 1991.

[21] K. Strobl and G. Hirzinger, "Optimal hand-eye calibration," in *IEEE International Conference on Intelligent Robots and Systems*, 11 2006, pp. 4647 – 4653.

[22] S. Gwak, J. Kim, and F. Park, "Numerical optimization on the euclidean group with applications to camera calibration," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 1, pp. 65–74, 2003.

[23] M. Ulrich and M. Hillemann, "Generic hand–eye calibration of uncertain robots," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11 060–11 066.

[24] Y. Wang, W. Jiang, K. Huang, S. Schwertfeger, and L. Kneip, "Accurate calibration of multi-perspective cameras from a generalization of the hand-eye constraint," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 1244–1250.

[25] M. Antonello, S. Ghidoni, and E. Menegatti, "Autonomous robotic system for thermographic detection of defects in upper layers of carbon fiber reinforced polymers," in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 2015, pp. 634–639.

[26] P. J. Schilling, B. R. Karedla, A. K. Tatiparthi, M. A. Verges, and P. D. Herrington, "X-ray computed microtomography of internal damage in fiber reinforced polymer matrix composites," *Composites Science and Technology*, vol. 65, no. 14, pp. 2071–2078, 2005.

[27] M. J. Emerson, K. M. Jespersen, A. B. Dahl, K. Conradsen, and L. P. Mikkelsen, "Individual fibre segmentation from 3d x-ray computed tomography for characterising the

fibre orientation in unidirectional composite materials," *Composites Part A: Applied Science and Manufacturing*, vol. 97, pp. 83–92, 2017.

[28] S. Zambal, W. Palfinger, M. Stöger, and C. Eitzinger, "Accurate fibre orientation measurement for carbon fibre surfaces," *Pattern Recognition*, vol. 48, no. 11, pp. 3324–3332, 2015.

[29] F. Giudiceandrea, E. Ursella, and E. Vicario, "A high speed ct scanner for the sawmill industry," in *Proceedings of the 17th international non destructive testing and evaluation of wood symposium*. University of West Hungary Sopron, Hungary, 2011, pp. 14–16.

[30] S. M. Stängle, F. Brüchert, A. Heikkila, T. Usenius, A. Usenius, and U. H. Sauter, "Potentially increased sawmill yield from hardwoods using x-ray computed tomography for knot detection," *Annals of forest science*, vol. 72, no. 1, pp. 57–65, 2015.

[31] J. A. Noble, R. Gupta, J. Mundy, A. Schmitz, and R. I. Hartley, "High precision x-ray stereo for automated 3d cad-based inspection," *IEEE Transactions on Robotics and automation*, vol. 14, no. 2, pp. 292–302, 1998.

[32] M. Antonello, M. Munaro, and E. Menegatti, "Efficient measurement of fibre orientation for mapping carbon fibre parts with a robotic system," in *International Conference on Intelligent Autonomous Systems*. Springer, 2016, pp. 757–769.

[33] D. Evangelista, M. Terreran, A. Pretto, M. Moro, C. Ferrari, and E. Menegatti, "3d mapping of x-ray images in inspections of aerospace parts," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 1223–1226.

[34] S. Tonello, G. P. Zanetti, M. Finotto, R. Bortoletto, E. Tosello, and E. Menegatti, "Workcellsimulator: A 3d simulator for intelligent manufacturing," in *Simulation, Modeling, and Programming for Autonomous Robots*, I. Noda, N. Ando, D. Brugali, and J. J. Kuffner, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 311–322.

[35] A. S. Glassner, Ed., *An Introduction to Ray Tracing*. GBR: Academic Press Ltd., 1989.

[36] A. Mavrinac, X. Chen, and Y. Tan, "Coverage quality and smoothness criteria for online view selection in a multi-camera network," *ACM Trans. Sen. Netw.*, vol. 10, no. 2, Jan. 2014. [Online]. Available: https://doi.org/10.1145/2530373

[37] K. Koide and E. Menegatti, "General robot-camera synchronization based on reprojection error minimization," in *Joint ARW & OAGM Workshop*, 2019.

[38] K. Shoemake, "Animating rotation with quaternion curves," *SIGGRAPH Comput. Graph.*, p. 245–254, 1985.

[39] K. Crane, M. Desbrun, and P. Schröder, "Trivial connections on discrete surfaces," *Computer Graphics Forum (SGP)*, vol. 29, no. 5, pp. 1525–1533, 2010.

[40] D. Evangelista, D. Allegro, M. Terreran, A. Pretto, and S. Ghidoni, "An unified iterative hand-eye calibration method for eye-on-base and eye-in-hand setups," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2022, pp. 1–7.

[41] G. Schweighofer and A. Pinz, "Globally optimal o (n) solution to the pnp problem for general camera models." in *BMVC*, 2008, pp. 1–10.

[42] S. Agarwal, K. Mierle, and T. C. S. Team, "Ceres Solver," 3 2022. [Online]. Available: https://github.com/ceres-solver/ceres-solver

[43] J. Miseikis, K. Glette, O. J. Elle, and J. Torresen, "Automatic calibration of a robot manipulator and multi 3d camera system," in *2016 IEEE/SICE International Symposium on System Integration (SII)*, 2016, pp. 735–741.

[44] D. Evangelista, E. Olivastri, D. Allegro, E. Menegatti, and A. Pretto, "A graph-based optimization framework for hand-eye calibration for multi-camera setups," 2023.

[45] F. Dornaika and R. Horaud, "Simultaneous robot-world and hand-eye calibration," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 4, pp. 617–622, 1998.

[46] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in *Numerical analysis*. Springer, 1978, pp. 105–116.

[47] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g 2 o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3607–3613.

Acknowledgments

This thesis work has been mainly supported by the SPIRIT project. The SPIRIT project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 779431. All the industrial partners of the *SPIRIT* project have been actively involved in making the inspection framework proposed in SPIRIT a successful project, for this reason, the author kindly acknowledges all the partners, in detail: *Marposs S.p.A., FACC Gmbh, Centro Ricerche Fiat S.C.P.A., InfraTec GmbH* and *BÖHLER Aerospace GmbH*.

I would like to send a very personal and special thanks to all the research staff of the *Intelligent Autonomous System Laboratory (IAS-Lab)* of the University of Padova. This mention goes particularly to *Emanuele* and *Alberto*, my two supervisors during the Ph.D. program, they supported me in all the research activities, giving suggestions and guiding my choices without imposing their subjective thought. This helped me to develop my own ideas on every problem and gave me the opportunity to grow up scientifically and personally. Special thanks have to be given to all the other researchers of the laboratory because they contributed to creating a very special and comfortable place in which research ideas can be developed with an open-minded set and a true and positive share of opinions.

Let me also mention my colleague and friend *Marco* that sincerely assisted me in every activity, sharing thoughts, developing ideas to improve our company or simply assisting me while I was dealing with difficult periods. I strongly believe that the team we were able to build together over the years will grow up and create very special things in the future.

All the people at *FlexSight* and *IT+Robotics* also require a special acknowledgment, they every day contribute to developing a healthy and peaceful working environment, full of exchange of ideas and competent people with whom work days are, one after another, moments of growth and continuous relationship.

I would like to leave a special thought for my family, *Silvio, Domenica* e *Francesca*. You have always supported me at every stage of my life, including this Ph.D. I know that I can always count on you, and I am happy to be able, in my own small way, to give you those satisfactions and joys you deserve in return for all the comfort and support you always give to me.

Last but not least, let me make a special mention to my wife *Linda*. I want to express my heartfelt gratitude to you, my dear wife. Your constant love, care, and unwavering support have

brought immense joy and comfort to my life, even during the most challenging times. As we look forward to the arrival of our first child, Flavia, I am filled with an overwhelming sense of gratitude for the greatest gift you have given me since we first met. Thank you for being the most amazing partner and mother-to-be. I love you.