robotics

MDPI

Article

Visual Servoing Architecture of Mobile Manipulators for Precise Industrial Operations on Moving Objects

Javier González Huarte * and Aitor Ibarguren

Industry and Transport Division, TECNALIA, Basque Research and Technology Alliance (BRTA), 20009 Donostia-San Sebastian, Spain; aitor.ibarguren@tecnalia.com
* Correspondence: javier.gonzalezhuarte@tecnalia.com

Abstract: Although the use of articulated robots and AGVs is common in many industrial sectors such as automotive or aeronautics, the use of mobile manipulators is not widespread nowadays. Even so, the majority of applications separate the navigation and manipulation tasks, avoiding simultaneous movements of the platform and arm. The capability to use mobile manipulators to perform operations on moving objects would open the door to new applications such as the riveting or screwing of parts transported by conveyor belts or AGVs. This paper presents a novel position-based visual servoing (PBVS) architecture for mobile manipulators for precise industrial operations on moving parts. The proposed architecture includes a state machine to guide the process through the different phases of the task to ensure its correct execution. The approach has been validated in an industrial environment for screw-fastening operations, obtaining promising results and metrics.

Keywords: visual servoing; mobile manipulation; screwing; industrial application

Citation: González Huarte, J.; Ibarguren, A. Visual Servoing Architecture of Mobile Manipulators for Precise Industrial Operations on Moving Objects. *Robotics* **2024**, *13*, 71. https://doi.org/10.3390/robotics13050071

Academic Editors: Andrius Dzedzickis and Hong Zhang

Received: 15 March 2024
Revised: 19 April 2024
Accepted: 29 April 2024
Published: 2 May 2024

1. Introduction

The use of robots is widely extended in the industry nowadays, making use of articulated robot arms [1,2] for a huge range of applications. Additionally, AGVs [3,4] have become popular for internal logistics during the last decades in different industries such as automotive or manufacturing sectors. In recent years, the appearance of mobile manipulators in the market has changed the perspective of users and developers, as this kind of robot extends the manipulation capabilities of articulated industrial arms to the whole workshop area. Even so, the current industrial applications with mobile manipulators usually separate the navigation and manipulation phases due to technical simplification as well as safety reasons [5,6]. For example, in many logistics applications, the platform initially navigates to the pickup point and grasps the object once the platform is still, which is a straightforward approach for a wide range of scenarios.

Nevertheless, the ability to simultaneously coordinate both platform and arm movements during operation further exploits the capacities of mobile manipulators. The capability to synchronize movement and operation not only enhances their functionality in stationary conditions but also enables operations in dynamic environments like the manipulation of objects that are being transported either by conveyor belts or AGVs. This feature is essential in many sectors, such as automotive or industrial manufacturing, where parts are constantly moved between stations. Although recent advances allow synchronous movement and manipulation in both static and dynamic environments, the ability to carry out high-precision tasks such as riveting or screwing in movement raises a challenging topic involving the coordination of robotic platform and arm movements while ensuring low positioning errors. Therefore, a practical and reliable implementation of this kind of technology requires including mechanisms to adapt to objects that are moving at an unknown velocity around the workshops, facilitating the fast deployment of mobile manipulators in different operations.

This paper proposes a novel position-based visual servoing (PBVS) architecture for mobile manipulators focused on precise industrial operations on moving parts, using a state machine to guide the process through its different phases. To ensure safe and precise contact between the robot tool and the part, the architecture includes compliance to maintain the stability of the task during this critical phase of the process. The architecture has been tested on an automotive sector setup in screw-fastening operations, obtaining promising results and metrics.

2. Related Work

Visual servoing [7–9] is a recurrent topic in robotics, as it provides the capability to control a robot's behavior based on visual feedback, adapting to the target's motion. Compared to a dynamic look-and-move approach, where robot movements or trajectories are generated using the information of one or a few images, visual servoing enables a continuous adaptation of the robot position required by industrial applications where the targets are moving.

Several works applied the visual servoing paradigm in industrial environments. Lippiello et al. [10] presented a position-based visual servoing approach for a multiarm robotic cell equipped with a hybrid eye-in-hand/eye-to-hand multicamera system which includes mechanisms for fast and accurate estimation as well as occlusions handling. The work includes two case studies, coordinated dual-arm movements and grasping tasks, for testing its effectiveness. For example, Castelli et al. [11] propose a novel visual servoing system based on machine learning for the automation of the winding of copper wire during the production of electric motors. It includes a Gaussian mixture model (GMM) machine learning system, which guarantees an extremely fast response in a path-following application simulating the real stator teeth of a generic electric motor. Additionally, Wu et al. [12] use a position-based visual servoing approach for pick-and-place operations in the food industry, which has been validated in a lab-scale experimental setup.

Focusing on precise industrial operations, Pitipong et al. [13] propose a visual servoing strategy for determining and correcting the alignment of an automated robot screw-fastening operation on a 4-DOF robot. Hao and Xu [14] also present an image-based visual servoing algorithm for screw grasping and assembling, adding a robotic assembly workflow to guide the process. Even so, sparse applications are found for applying visual servoing in mobile manipulators for precise industrial tasks.

The addition of state machines to visual servoing is an interesting topic, as it supports guiding the addressed task. Ruybal et al. [15] propose the use of a finite state machine to increase the positional accuracy of a robotic arm under impedance control in peg-in-hole insertion tasks. Aref et al. [16] also make use of a state machine to solve logical control issues on a pallet-picking application using a forklift truck. Additionally, Fue et al. [17] propose the use of finite-state machines and visual servoing in a cotton-harvesting platform. The different works show how state machines can enhance visual servoing controllers in complex tasks composed of different process steps, although its use is not very extended.

Finally, many works on visual servoing for mobile manipulators follow an image-based approach with an eye-in-hand configuration. For example, Wang et al. [18] propose an image-based visual servoing approach which is enhanced with a Q-learning controller through a rule-based arbitrator. Belmonte et al. [19] present a direct image-based controller taking into account robot dynamics during the tracking of images. Li et al. [20] make use of an HVS control method combined with a Kalman Filter to grasp static objects, obtaining a flexible and reliable grasping effect. Arora and Papachristos [21] tackle the problem of the real-time tracking and grasping of a dynamic target by a mobile manipulation robot, inspired by the principles of visual servoing and the paradigm of virtual force-based formulation. Finally, Burgess-Limerick et al. [22] pose a generalized architecture for reactive mobile manipulation for grasping objects with unpredictable motion, tackling both scenarios of static and dynamic objects.

The above-mentioned papers address the tracking [19] and grasping of static [18,20] and dynamic objects [21,22] with mobile manipulators. Even so, further research is needed to meet the requirements of industrial applications involving the manipulation of moving objects. Operations such as assembling or screw fastening require high accuracy and reliability, crucial aspects in assembly lines of automotive or manufacturing industries.

3. Proposed Approach

The paper proposes a visual servoing architecture for mobile manipulators, focusing on the capability to perform precise operations on moving objects in industrial scenarios. During the design of the architecture, several considerations related to the characteristics of industrial scenarios are taken into account:

- The mobile manipulator should be able to approach the moving object and perform different operations on the same parts (e.g., tighten five different bolts). An eye-to-hand configuration is proposed with the camera attached to the mobile platform. This decision is motivated by the fact that in real applications, it could be problematic to find a suitable camera position that ensures the visibility of the part in a wide range of tool positions. The eye-to-hand configuration simplifies the problem, making it necessary to detect just one part instead of multiple small elements such as bolts or holes.
- To generalize the approach and as it is intended for a wide range of parts, a position-based visual servoing (PBVS) is proposed to avoid conditioning the detection process.
- Due to the precision required for tasks such as screwing or riveting, the architecture includes a compliance layer to ensure that imprecisions do not damage tools or parts during contact phases.
- The industrial operations tackled in this paper require a series of steps for completion (e.g., an initial approach, perpendicular approximation to the bolt/hole, tool activation, safe tool retracting, and moving to the next operation). Consequently, the architecture should include mechanisms to guide the visual servoing process along the different steps of the procedure, parametrizing the control parameters and poses based on the current phase. The use of a state machine is proposed to this end.

Based on these premises, the paper proposes a visual servoing control architecture for mobile manipulators, where the key element is the state machine that guides the whole task, modifying the control parameters based on the current process phase. Additionally, the architecture also includes compliance to ensure stability during the phases where the robot tool and the part are in contact.

4. Notation and Reference Frames

This paper adopts specific notation and reference frames to describe the kinematics involved. The following key terms and frames of reference are outlined:

- Reference frames a and b are represented as $\mathcal{F}_a$ and $\mathcal{F}_b$, respectively.
- The translation vector of frame $\mathcal{F}_a$ relative to frame $\mathcal{F}_b$ is given by $\mathbf{p}_{ba} \in \mathbb{R}^3$.
- The rotation of frame $\mathcal{F}_a$ relative to frame $\mathcal{F}_b$ is represented by a rotation matrix $\mathcal{R}_{ba} \in SO(3)$.
- Consequently, the transformation matrix that defines the rigid transformation of frame $\mathcal{F}_a$ relative to frame $\mathcal{F}_b$ is denoted by $\mathbf{H}_{ba} \in SE(3)$, incorporating both translation and rotation components, and is expressed as

$$H_{ba} = \begin{bmatrix} R_{ba} & p_{ba} \\ 0 & 1 \end{bmatrix}. \tag{1}$$

- A twist vector $V = \begin{bmatrix} \omega & v \end{bmatrix}^T \in \mathbb{R}^6$ represents a rigid body motion. It comprises angular velocity $\omega \in \mathbb{R}^3$ and linear velocity $v \in \mathbb{R}^3$.

– The Jacobian matrix $J(\theta) \in \mathbb{R}^{6\times n}$ relates the joint velocities to the end-effector twist V_e, where $\theta \in \mathbb{R}^n$ represents the vector of joint positions, and n denotes the number of joints of the robot arm.

Furthermore, a variety of coordinate system frames that correspond to the mobile manipulator and the target object are depicted in Figure 1. Particularly, the following frames are employed in the paper:

– F_b represents the mobile platform base frame, located at the center of the mobile platform's base.
– F_a stands for the arm's base frame, mounted on top of the mobile platform.
– F_c corresponds to the camera frame.
– F_e designates the end-effector's frame, positioned at the extremity of the tool.
– F_o refers to the object frame, corresponding to the object whose motion is tracked.
– F_t indicates the target operation pose, located on the moving object.

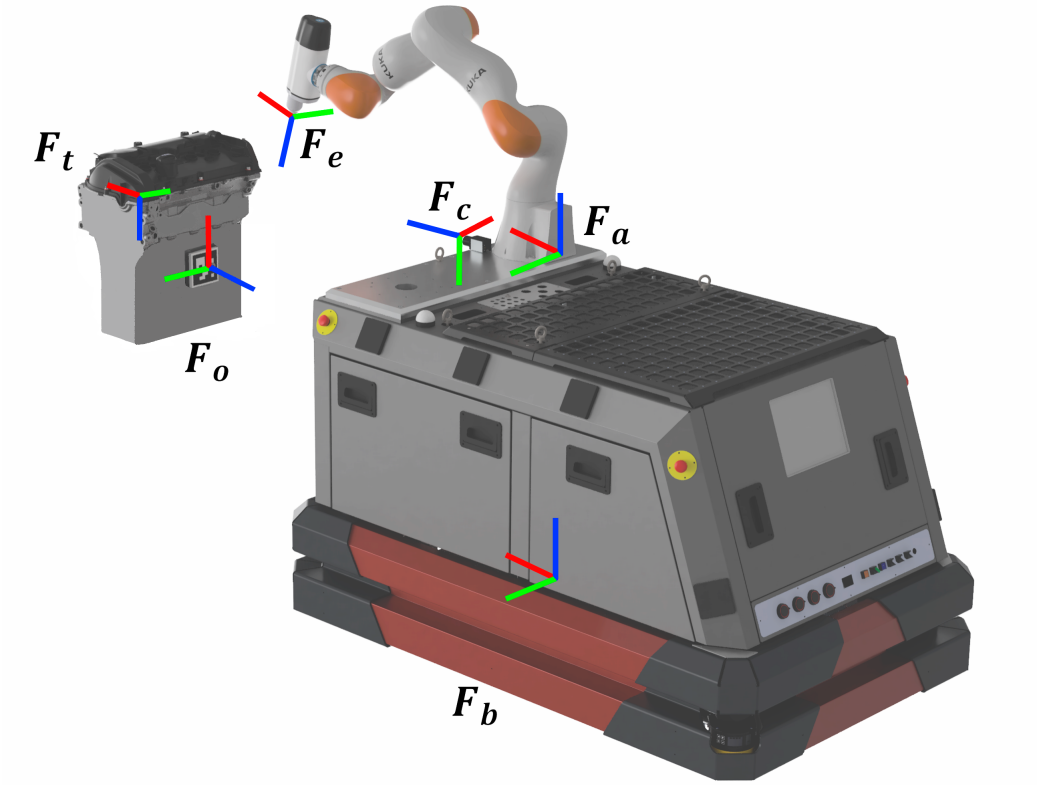

Figure 1. Mobile manipulator reference frames.

5. System Architecture

A visual servoing architecture is proposed for mobile manipulator control for industrial operations on moving objects with an eye-to-hand configuration. The approach assumes the knowledge of the geometric information of the mobile manipulator. Specifically, homogeneous transformation matrices

$$H_{ba}, \tag{2}$$

$$H_{bc}, \tag{3}$$

where H_{ba} defines the transformation between the mobile platform base and the arm and H_{bc} represents the transformation between the mobile platform base and the camera mounted on the platform.

Based on these premises, the architecture shown in Figure 2 is composed of seven different modules. The next lines summarize the function of each of these components.

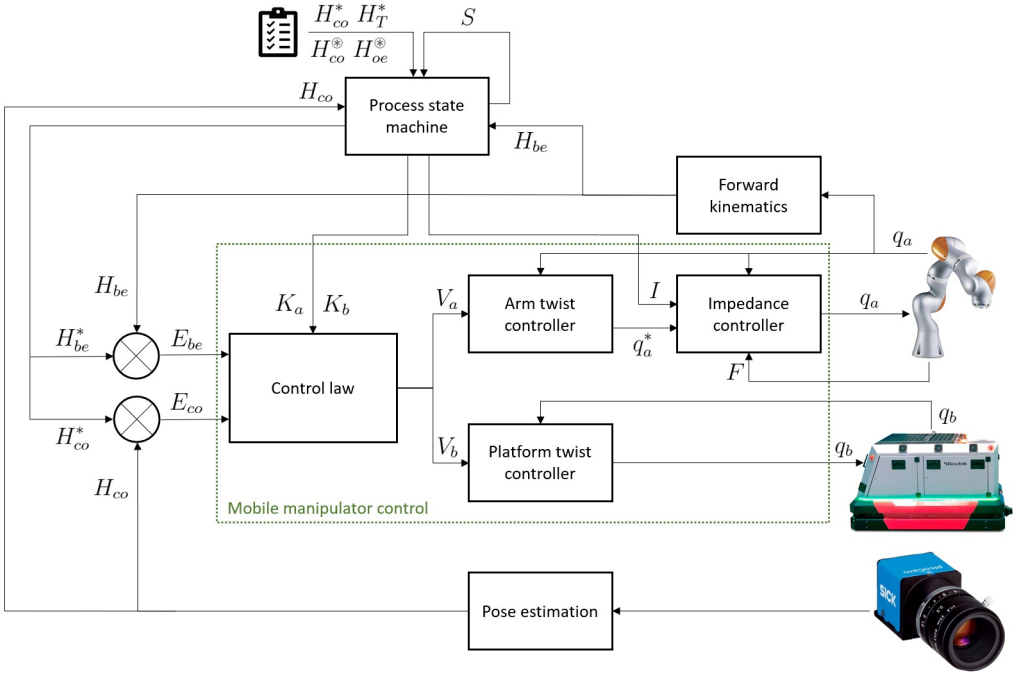

Figure 2. Schema of the mobile visual servoing architecture.

- The *pose estimation* module provides the transformation H_{co} of the object relative to the camera frame based on the received image.
- The *forward kinematics* module provides the transformation H_{be} between the mobile platform base and the end effector of the tool based on the joint positions q_a of the manipulator.
- The high-level management of the whole process is performed by the *process state machine*. Its main inputs are the transformation H^*_{co}, defining the target object position in the camera frame, as well as a set of transformations H^*_{bt} defining the operation poses in the object's frame (e.g., screw positions on the part). Additionally, input poses $H^{\circledast}_{be}$ and $H^{\circledast}_{co}$ define the final end effector and camera poses that the mobile platform should reach when it moves away from the part. Based on this input and the current state S of the state machine, the module manages the desired camera pose H^*_{co}, desired end-effector pose H^*_{be}, and PID values K_a and K_b sent to the *control law*, alongside the compliance values I sent to the *impedance controller*. Thus, the state machine can manage the different control parameters through the complete task, tuning the values to the requirements of each step.
- The *control law* module is in charge of generating the twist commands sent to the mobile manipulator based on the measured errors. Specifically, it receives both the end-effector error E_{be} and camera error E_{co}, as well as the PID values K_a and K_b for the arm and mobile platform, respectively. The module calculates the next twist vectors V_a and V_b for both the manipulator and mobile platform.
- Once the manipulator's twist vector V_a is calculated, the *arm twist controller* generates the next desired joint positions q^*_a that satisfy the motion required by the twist vector V_a.
- Lastly, the joint positions q^*_a are modified by the *impedance controller* using the compliance values I, calculating the final joint positions q_a.
- In parallel, the platform twist controller calculates the next mobile platform control command q_b based on the received Cartesian twist vector V_b.

While the preceding lines briefly explain how the flow of the proposed architecture works, the next sections provide further details about the *process state machine*, as well as the different modules composing the *mobile manipulator control*.

6. Process State Machine

As stated in the previous section, the process state machine acts as a supervisor of the whole operation and modifies different control parameters along the task. The main motivation to include such an element is that industrial operations like screwing or riveting are composed of many steps that require careful decision-making and management (e.g., approaching the operation area, approximating the task point perpendicularly, ensuring contact during the operation, retracting the tool safely, and continuing to the next operation poses). Therefore, a classical approach where a single target is provided to the control system is not sufficient to cope with the complexity of real industrial operations. The inclusion of a state machine tries to overcome this limitation, adding a high-level supervisor that ensures that the whole operation is fulfilled stably and safely. Specifically, the proposed state machine manages a total of eight states during the process, included in the *operate while moving* sub-state machine, plus four more high-level states devoted to process stability. The current process state S is defined as

$$S \in \{s_1, s_2, \ldots, s_8\}, \tag{4}$$

where each state describes a step of the whole operation. The state machine transits between these states when some requirements are fulfilled, mainly related to reaching some error tolerances. For convenience, during the description of the module, some states are grouped into a sub-state machine to facilitate its understanding.

Initially, the state machine receives several input parameters defining the key information of the task. Precisely, there are four input parameters related to the task-specific positioning strategy, parameters that remain static during the whole task. The principal ones are the target pose between the camera and the object

$$H_{co}^{*}\,, \tag{5}$$

and a set of poses H_T^* on the object frame describing the N positions the tool should reach to fulfill the task, defined as

$$H_T^* = \left\{H_{ot_1}^*, H_{ot_2}^*, \ldots, H_{ot_N}^*\right\}. \tag{6}$$

Additionally, two more poses define a safe retracting maneuver of the robot once the task is finished, moving away both the manipulator and mobile platform from the part avoiding any undesired collision. Specifically, these poses are defined as

$$H_{co}^{\circledast}\,, \tag{7}$$

$$H_{oe}^{\circledast}\,, \tag{8}$$

where $H_{co}^{\circledast}$ defines the final camera pose, and $H_{oe}^{\circledast}$ determines the final end-effector pose in the part's frame.

Additionally, several control parameters are verified and modified during operation by the state machine according to the current state. These parameters are mostly those related to the control cycle and the low-level control of the system.

- In each control cycle, the input parameters H_{co}^* and H_T^* are compared with the current camera pose H_{co} and end-effector pose H_{be} to check if the error is within the tolerance, so the process can advance to the next state. In particular, during the platform approach, the error of the camera's pose is verified to manage the transition from state s_i as

$$S = \begin{cases} s_{i+1} & \text{if } |p_{cc^*}| < \epsilon_t \ \wedge \ |\theta_{cc^*}| \leq \epsilon_r \\ s_i & \text{otherwise} \end{cases}, \tag{9}$$

where $|p_{cc^*}|$ and $|\theta_{cc^*}|$ denote the translation and rotation error of the camera pose calculated as

$$H_{cc^*} = H^*_{co} \,.\, H^{-1}_{co}, \tag{10}$$

$$|\theta_{cc^*}| = arccos\left(\frac{tr(R_{cc^*}) - 1}{2}\right) \tag{11}$$

and ϵ_t and ϵ_r define the translation and rotation threshold.
Alternatively, during the phases where the manipulator moves around the operation points, the transition state s_i is managed as

$$S = \begin{cases} s_{i+1} & \text{if } |p_{be^*}| < \epsilon_t \ \wedge \ |\theta_{be^*}| \leq \epsilon_r \\ s_i & \text{otherwise} \end{cases}, \tag{12}$$

where $|p_{be^*}|$ and $|\theta_{be^*}|$ denote the translation and rotation error of the end effector

$$H_{be^*} = H^*_{be} \,.\, H^{-1}_{be}, \tag{13}$$

$$|\theta_{be^*}| = arccos\left(\frac{tr(R_{be^*}) - 1}{2}\right) \tag{14}$$

and ϵ_t and ϵ_r define the translation and rotation threshold.

- Based on the current process state, the module modifies the different control parameters to ensure that the values are appropriate for the characteristics of each process phase. In particular, the next parameter sets are managed:
  - The target pose of the end effector H^*_{be} is modified during the different operation phases for two main purposes. The first one is to iterate along the different target poses in H^*_T as the task involves operating on several elements. The second one is to add a safe approach and retract the pose before and after the operation; the main motivation is to ensure that the tool enters perpendicularly to the operation pose, avoiding any unsafe movement near the part.
  - The PID parameters of the control law are also modified by the state machine. In each cycle, the module provides a set of PID parameters for the manipulator as

$$K_a = \begin{bmatrix} K_{p_a} \\ K_{i_a} \\ K_{d_a} \end{bmatrix}, \tag{15}$$

    where K_{p_a}, K_{i_a}, and K_{d_a} define the PID values for the manipulator and a set of PID parameters for the mobile platform as

$$K_b = \begin{bmatrix} K_{p_b} \\ K_{i_b} \\ K_{d_b} \end{bmatrix}, \tag{16}$$

    where K_{p_b} , K_{i_b}, and K_{d_b} define the PID values for the mobile platform.
    During the operation process, the PID parameters of the mobile platform are boosted in the approach steps to speed up the process, denoted by $K_{b_\triangle}$, while the proportional part decreases during the operation phases to ensure precision, represented by $K_{b_\triangledown}$. Therefore, K_b is defined as

$$K_b \in \left\{K_{b_\triangle}, K_{b_\triangledown}\right\}. \tag{17}$$

The manipulator also includes these approach and operation parametrizations, denoted by $K_{a_\triangle}$ and $K_{a_\triangledown}$, as well as a zero vector, denoted as $K_{a\varnothing}$, to avoid any arm movement during some steps of the process. Thus, K_a is defined as

$$K_a \in \left\{K_{a_\triangle}, K_{a_\triangledown}, K_{a\varnothing}\right\}. \tag{18}$$

– Finally, the state machine also modifies the impedance parameters I defining the compliance of the manipulator. The impedance parameters are defined as

$$I = \{K, D, M\}\,, \tag{19}$$

where K, D, and M describe diagonal matrices defining the stiffness, damping, and mass values. In particular, this module manages two different compliant profiles, a high-stiffness profile denoted by I_s where the arm's position is completely rigid, and a compliant profile denoted by I_c where the manipulator shows compliant behavior in the X, Y, and Z axes. Thus, I is defined as

$$I \in \{I_s, I_c\}. \tag{20}$$

As a general rule, the high-stiffness profile I_s is used when there is no possible contact with the part, while the compliant profile I_c is used when the tool is in contact with the part or is about to be. The decision to include the high-stiffness profile is to avoid deviations from the nominal path during the initial approaches due to imprecision in the force/torque sensors, which may lead to undesired behaviors during contact between tool and part (e.g., error during insertion of the tool on the part).

Summarizing, the process state machine modifies and sends these control parameters to other modules of the architecture to guide the different phases of the operation.

6.1. State Machine

At the highest level, the state machine manages the system's stability to ensure that the process stops safely in case the target is lost for a long time, as shown in Figure 3. Specifically, the process is managed through the following states.

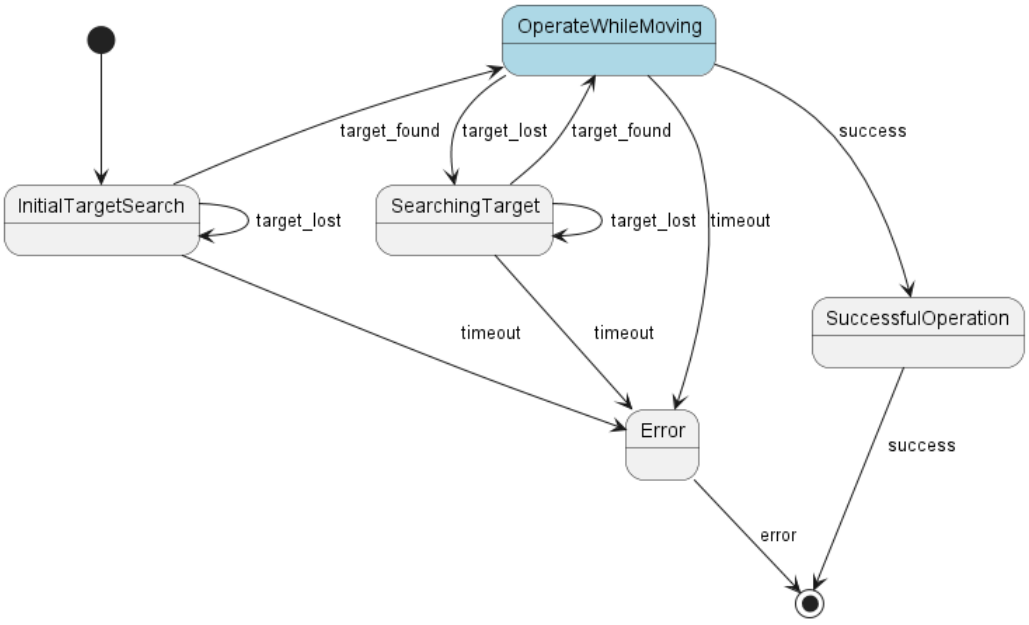

Figure 3. Complete state machine.

- **Initial target search:** The process starts with the mobile manipulator waiting for the target object to be detected. If the target is detected, the process moves to the *operate while moving* sub-state machine. Otherwise, if the target detection timeout is reached, the process transits to the *error* state.
- **Operate while moving:** This sub-state machine contains the complete operation sequence that is explained later in this section, states s_1 to s_8. In case the target is lost during the process, the process moves to the *searching target* state. Additionally, if any error is detected during the operation, the process reaches the *error* state. Otherwise, once the task is completed, the *successful operation* state is reached.
- **Searching target:** Once the target is lost, this state stops any movement and tries to search for the target again. If the target is found again within a time limit, the state machine returns to the previous state in the *operate while moving* sub-state machine. If the timeout is reached, the process transits to the *error* state.
- **Error:** In this state, the process is terminated in a controlled way, triggering any error management before exiting the state machine.
- **Successful operation:** This state terminates the process.

The next paragraphs describe in depth the *operate while moving* sub-state machine presented above.

6.2. Operate While Moving

The *operate while moving* sub-state machine, as mentioned, manages the actual operation while the object is being detected. The operation is managed through a series of states that go from s_1 to s_8 as depicted in Figure 4.

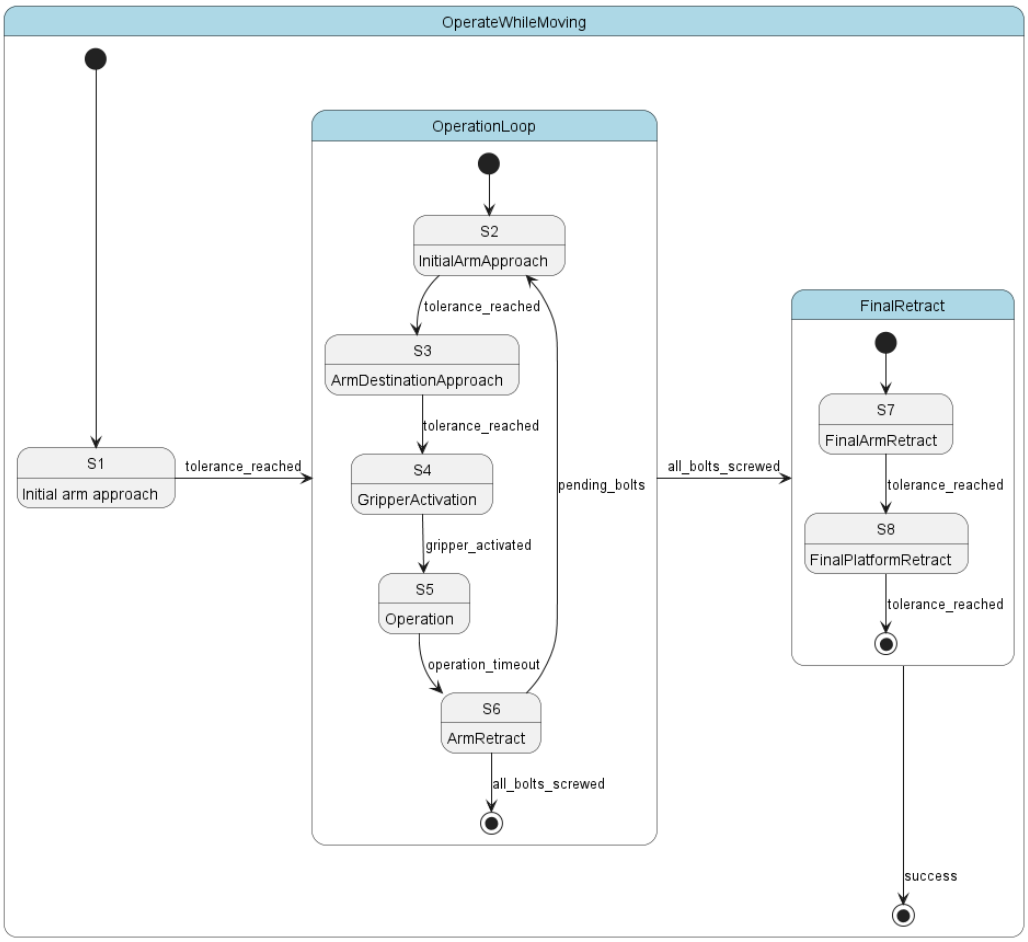

Figure 4. *Operate while moving* sub-state machine.

The complete task is divided into three different phases:

- **s_1—Initial platform approach:** In this initial state s_1, the mobile platform approaches the target until it is near enough to start the manipulation task. For the control parameters, the PID values for the mobile platform are boosted with values $K_{b_\triangle}$ while the arm's ones are set to zero value $K_{a_\varnothing}$ to avoid any manipulator movement. The state transition is managed by checking that the camera is close to the target object's position using Equation (9). If the errors are below the threshold, the state transits to state s_2.

Once the mobile platform reaches the desired camera pose and follows the target part, the state machine moves to the complete manipulation operation. As mentioned previously, the operation is guided by the task poses H_T^* where the manipulator activates the tool. Even so, to ensure the security of the operation an additional pose is introduced above each task pose; see Figure 5, which allows to approach and retract safely. Therefore, the state machine modifies the desired end-effector pose H_{be}^* to introduce the aforementioned maneuver.

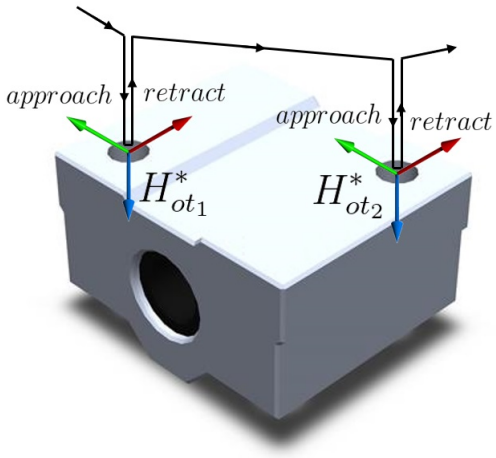

Figure 5. Approach and retract maneuvers to operation poses.

Specifically, the next states guide the manipulation procedure through the whole sequence:

- **s_2—Initial arm approach:** In this initial approach state s_2, the manipulator's set-point H_{be}^* is set to a pose that enables a safe maneuver towards the operation point, calculated as

$$H_{be}^* = H_{bc} \,.\, H_{co} \,.\, H_{ot_i}^* \,.\, H_{tp} \,, \tag{21}$$

where $H_{ot_i}^*$ defines the i$^{\text{th}}$ operation pose and H_{tp} determines the transformation between the nominal operation pose and the approach pose.
Regarding the PID values, the mobile platform is set to $k_{b_\triangledown}$ operation values, while the manipulator is boosted to $K_{a_\triangle}$ values to accelerate the approach. Additionally, the manipulator is parametrized with I_s to show a high-stiffness behavior that ensures that the tool reaches the approach pose above the operation point, without any deviation due to compliance. The state transition is managed by the end-effector error using Equation (12).
- **s_3—Arm destination approach:** In state s_3, the manipulator moves towards the operation point. The setpoint H_{be}^* is calculated as

$$H_{be}^* = H_{bc} \,.\, H_{co} \,.\, H_{ot_i}^* \,, \tag{22}$$

where $H_{ot_i}^*$ defines the i$^{\text{th}}$ operation pose.
Both PID values are set to values $k_{a_\triangledown}$ and $K_{b_\triangledown}$ to prioritize stability over approach speed. At the same time, the manipulator is parametrized with I_c to show compliant

behavior to avoid any damage during contact. The state transition is managed by the end-effector error using Equation (12).

- s_4**—Gripper activation:** This state s_5 manages the gripper activation. As this activation can take some time depending on the type of tool used, it should be performed asynchronously. Therefore, the control loop continues in this state until the activation is finalized. The PID and compliance values are set to $k_{a_\triangledown}$, $k_{b_\triangledown}$, and I_c while the state machine moves to state s_5 when the gripper activation finishes.
- s_5**—Operation:** The operation state s_6 controls the operation time where the tool is working on the industrial task for a defined time t. Again, the PID and compliance values are set to $k_{a_\triangledown}$, $k_{b_\triangledown}$, and I_c while the state machine moves to state s_6 after time t.
- s_6**—Arm retract:** This last state s_7 manages a safe retract maneuver, moving the manipulator to the approach position. The manipulator's setpoint H^*_{be} is calculated as in Equation (22). Regarding the PID values, the mobile platform is set to $k_{b_\triangledown}$ operation values while the manipulator is boosted with $K_{a_\triangle}$ values to speed up this step. Finally, the manipulator is parametrized with compliant values I_c to ensure a safe tool removal.
  The state transition is calculated as in the previous approach maneuver using Equation (12).

These previous steps are repeated for each pose in H^*_T. Afterwards, the process concludes with the final retract maneuver that ensures a safe manipulator and mobile platform movement away from the part. As a first step, the manipulator retracts and once it is removed from the manipulation area, the platform abandons the part zone.

- s_7**—Final arm retract:** This state s_7 drives a safe retract maneuver, moving the manipulator away from the part. The manipulator's setpoint H^*_{be} is calculated as

$$H^*_{be} = H_{bc} \,.\, H_{co} \,.\, H^{\circledast}_{oe}\,, \tag{23}$$

  where $H^{\circledast}_{oe}$ defines the retract pose of the end effector in the object's frame.
  The PID and compliance values are set to $k_{a_\triangledown}$, $k_{b_\triangle}$, and I_s. The state transition is calculated as in the previous approach maneuvers using Equation (12).
- s_8**—Final platform retract:** Finally, state s_8 moves the mobile platform from the object area. The mobile platform setpoint H^*_{co} is set to

$$H^*_{co} = H^{\circledast}_{co}\,, \tag{24}$$

  where $H^{\circledast}_{co}$ defines the final object pose in the camera frame.
  The PID values of the mobile platform are boosted to $k_{b_\triangle}$ while the manipulator values are set to $K_{a\varnothing}$ to deactivate the arm's movements. The state transition is estimated by verifying the object's pose using Equation (9).

Based on this state machine, the module tunes and parametrizes the different poses and values of the *control law* and *impedance controller* modules.

7. Mobile Manipulator Control

As presented in the previous section, the process state machine is responsible for guiding the complete task and tuning the different control parameters along the whole process. On a lower level, the modules composing the mobile platform control generate the manipulator and mobile platform commands based on the current error. The next lines describe the internal calculus of the different modules.

7.1. Control Law

The *control law* module calculates the twist vectors V_a and V_b of both the robotic manipulator arm and mobile platform based on the received errors. As mentioned previously, the process state machine is the high-level module that modifies and tunes the control

parameters of the *control law* module, providing a dynamic behavior along the different steps of the task.

Initially, the twist vector V_b of the mobile platform is calculated. For this, the error E_{co}, represented on the camera's frame, is transformed to be represented on the base frame using

$$H_{bE_o} = \begin{bmatrix} R_{bc} & 0 \\ 0 & 1 \end{bmatrix} . E_{co} \, , \tag{25}$$

where R_{bc} defines the rotational part of the transformation between the robot base and the camera.

This error matrix is converted into vector form L_{E_o} for convenience in further steps as

$$L_{E_o} = \begin{bmatrix} \omega_{E_o} \\ \nu_{E_o} \end{bmatrix}, \tag{26}$$

where ω_{E_o} and ν_{E_o} represent the angular and linear parts of the error, successively. For the angular subvector ω_{E_o}, the rotation part of the error matrix H_{bE_o} is transformed into angle-axis representation [23] as

$$\theta = \arccos \frac{r_{11} + r_{22} + r_{33} - 1}{2}, \tag{27}$$

$$a_1 = \frac{r_{32} - r_{23}}{2 \sin \theta}, \tag{28}$$

$$a_2 = \frac{r_{13} - r_{31}}{2 \sin \theta} \tag{29}$$

$$a_3 = \frac{r_{21} - r_{12}}{2 \sin \theta} \tag{30}$$

where r_{xy} denotes the element on row x and column y in the rotation matrix R_{bE_o}. These values are then used to define the concise rotation vector ω_{E_o} as

$$\omega_{E_o} = \begin{bmatrix} \theta a_1 \\ \theta a_2 \\ \theta a_3 \end{bmatrix}. \tag{31}$$

The linear vector ν_{E_o} can be directly extracted from the translational component p_{bE_o} of H_{bE_o} as

$$\nu_{E_o} = p_{bE_O}. \tag{32}$$

This error vector L_{E_o} is used to calculate the twist vector V_b of the mobile platform, applying PID control [24] as

$$V_b = K_{p_b} L_{E_o} \; + \; K_{i_b} \int_0^t L_{E_o} \, dt \; + K_{d_b} \frac{dL_{E_o}}{dt} \tag{33}$$

where K_{p_b}, K_{i_b}, and K_{d_b} are diagonal matrices with the proportional, integral, and derivative values of the mobile platform.

Once the twist vector V_b of the mobile platform is calculated, the arm's twist V_a is computed subsequently. To avoid overcompensation associated with the combined motions of both the platform and the arm, it is important to determine the impact that the platform's motion, represented by the twist vector V_b, exerts over the end-effector motion [25]. This exerted motion is represented by the twist vector V_{b_e} as

$$V_{b_e} = \begin{bmatrix} \omega_b \\ \nu_b + r \times \omega_b \end{bmatrix}, \tag{34}$$

where ν_b and ω_b represent the translation and rotation part of the twist vector V_b, and r defines the translation between the mobile platform base and end effector. This twist vector V_{b_e} is further used to compensate for the end-effector error, using

$$E'_{be} = E_{be} - V_{b_e}\, dt. \tag{35}$$

This error E'_{be} is converted into its vector form L_{E_e} as presented previously in Equations (26)–(31). As a final step, twist vector V_a of the arm is calculated using, again, PID control as

$$V_a = K_{p_a} L_{E_e} + K_{i_a} \int_0^t L_{E_e}\, dt + K_{d_a} \frac{dL_{E_e}}{dt} \tag{36}$$

where K_{p_a}, K_{i_a}, and K_{d_a} are diagonal matrices with the proportional, integral, and derivative values for the robot arm.

These two twist vectors V_a and V_b of both the arm and mobile platform are further sent to the *arm twist controller* and *platform twist controller* for further management.

7.2. Arm Twist Controller and Compliance Controller

The *arm twist controller* and *compliance controller* are in charge of managing the arm twist vector V_a and moving the manipulator's joints. Specifically, the first module computes the next joint positions that ensure that the end effector moves at the desired velocity, while the second one allows a compliant behavior of the manipulator in the contact operations involved in the defined task.

The first step of the sequence is to convert the twist command V_a into the desired joint positions q_a^* that ensure the desired velocity command. To this end, the *arm twist controller* calculates the joint positions q_a^* as

$$q_a^* = q_a + J^+(q_a)\, V_a\, dt \tag{37}$$

where q_a is the current joint state and $J^+(q_a)$ defines the pseudo-inverse of the Jacobian matrix [26] of the manipulator, as the proposed approach is intended for manipulators up to seven degrees of freedom.

In a second step, the *compliance controller* module [27,28] allows establishing a mass-damper-spring relationship between the Cartesian position Δx and the Cartesian force F. To this end, the following formula is applied:

$$F = M\Delta\ddot{x} + D\Delta\dot{x} + K\Delta x, \tag{38}$$

where M, D, and K represent the virtual inertia, damping, and stiffness of the system, respectively.

Based on the sensed force vector F, the ΔX defining the displacement in Cartesian space is calculated as

$$\Delta X = \frac{\Delta F}{M\Delta t^2 + D\Delta t + K}, \tag{39}$$

where ΔF represents the difference between the desired contact force and the actual one.

This ΔX is added to the nominal joint positions q_a^* to provide the compliant behavior using

$$q_a = q_a^* + J^+(q_a^*)\, \Delta X, \tag{40}$$

where $J^+(q_a^*)$ defines the pseudo-inverse of the Jacobian matrix of the manipulator in the nominal joint positions q_A^*. With this last step, the next joint position q_a is calculated and sent to the manipulator, adding the compliant behavior that allows safe contact between the tool and part during the operation.

7.3. Platform Twist Controller

This module converts the twist command V_b of the mobile platform into the low-level commands q_b of the platform. As the proposed approach is intended for different platform types and geometries such as Ackermann steering [29] or omnidirectionality [30], this section will not provide any specific equations about the different mobile platform configurations.

8. Implementation

The proposed architecture is implemented for a screwing-while-moving operation, in which the mobile manipulator must fasten several screws on a part that is subject to unpredictable motion. Figure 6 shows the mobile manipulator employed in this implementation, which includes the following elements.

Figure 6. Mobile manipulator used for the implementation of the mobile visual servoing architecture.

- An omnidirectional mobile platform of 1.686 × 1.016 × 933 mm with mechanum wheels [31] able to move at a speed of up to 1.0 m/s. The platform includes two manipulators in a dual-arm configuration, although a single arm is used for the presented validation process.
- A Kuka LBR iiwa 7 [32] manipulator, an articulated robot arm of seven axes with torque sensors equipped in each joint.
- An industrial IDS [33] UI-5240CP monochrome camera with a resolution of 1280 × 1024 and a frequency of up to 50 fps. A LED lighting system is also included to ensure the illumination conditions and avoid detection errors due to changes in ambient light.
- For the screw-fastening task, the manipulator includes an OnRobot multifunctional screwdriver [34]. This screwdriver incorporates torque control as well as intelligent error detection for the management of the screwing process.
- An industrial PC is included on the mobile manipulator, which executes the different software modules implemented on the paper.

From the software point of view, all the modules are developed using the ROS framework [35]. Specifically, the presented architecture is composed of different ROS nodes offering multiple actions and services implemented in C++. The next lines provide details of several features of the software:

- The Kuka LBR iiwa manipulator control is implemented using the Direct Servo library. The control loop of the arm includes both the *arm twist controller*, which runs on

the industrial PC at a frequency of 250 Hz, and the *impedance controller* which is implemented on the robot controller.

- To simplify the detection and allow a high-frequency pose estimation, an Aruco marker detector library [36] is included. The detection provides a 6D pose at a frequency of 40 Hz based on the camera setup described previously.
- The *control law* is executed at a frequency of 40 Hz based on the input provided by the marker detector. Therefore, the twist commands are generated at a frequency of 40 Hz, although the low-level control of the manipulator manages the twist commands at 250 Hz, ensuring a smooth motion and compliance during the execution.

The presented architecture is implemented and further used for the validation process as described in the next section.

9. Validation

To validate the proposed approach, the architecture is tested on an industrial setup, where the mobile manipulator fastens the screws of a motor cover that is moving. Particularly, the task involves fastening four screws located on the upper part of the motor cover. During the task, the platform follows the motion of the motor cover, while the arm equipped with the screwdriver adheres to the bolts to carry out the fastening process as shown in Figure 7. As mentioned previously, an Aruco marker is attached to the motor frame to simplify the detection and allow a high-frequency pose estimation.

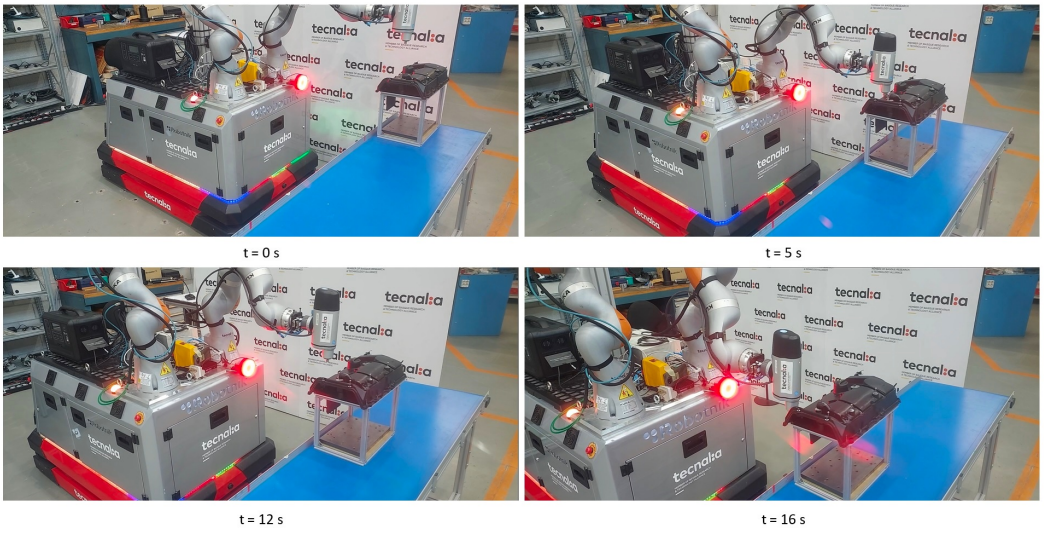

Figure 7. Mobile manipulator fastening screws of the moving motor cover.

To measure the performance of the architecture, the error of the camera E_{co} and end effector E_{be} is continuously monitored. Figure 8 shows the platform and arm error acquired during a complete sequence of four screw-fastening processes. The platform error, depicted in orange, falls in the initial platform approach phase and maintains stability throughout the whole process around a value of 20 mm. In the case of the arm error, depicted in blue, the graph exhibits three peaks in each screw related to the state transitions where the target poses are modified (approach pose, screw tightening pose, and retract pose). In these transitions, the error rises due to the change in the setpoint although it is quickly reduced during the next cycles.

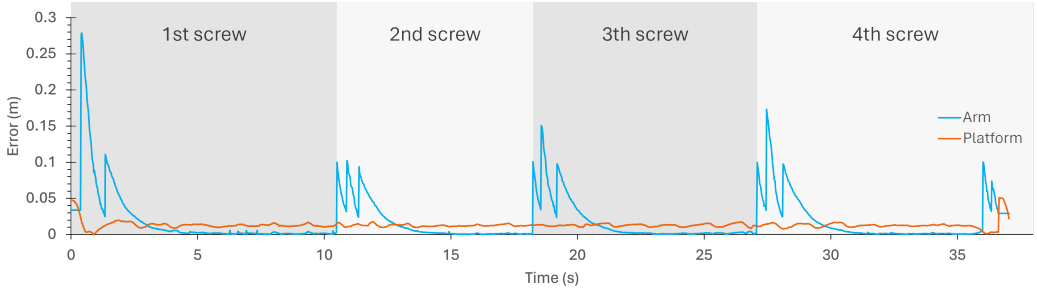

Figure 8. Camera error E_{co} and end-effector error E_{be} throughout a complete sequence of four screw-fastening processes.

For a deeper insight into the process, Figure 9 illustrates the error data of the first screw of the sequence, including the state value (in black) to highlight the transitions between phases. Initially, the platform approaches the motor cover on state 1, reducing the platform error until the transition threshold is reached. In state 2, the arm error initially exceeds 0.25 m; this error depends on the initial position of the arm, which can be far from the first screw. During this phase, the arm rapidly adjusts to the new target (the first screw's approach pose), evidenced by the steep decrease in error and reflecting the responsiveness of the arm. The screw approximation phase modifies again the setpoint to the screw pose, increasing the error momentarily. The arm rapidly adjusts again to the new target and maintains this low error during the operation phase, where the screw is fastened by the screwdriver. This setpoint modification pattern is systematically repeated on subsequent screw attempts, confirming the system's ability to adapt quickly to new positional targets.

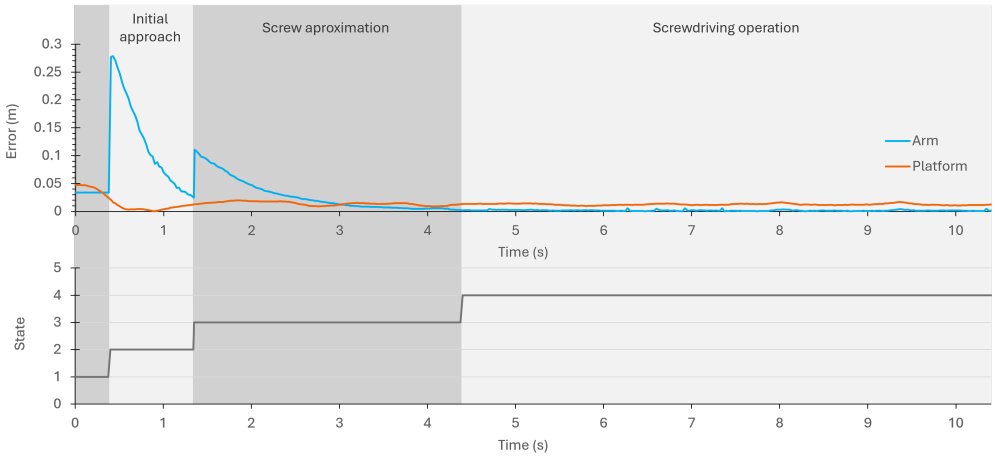

Figure 9. Camera error E_{co} and end-effector error E_{be} during a single screw operation throughout the initial approach, screw approximation, and operation states.

To further assess the performance of the system, the architecture is validated against two different scenarios:

- **Constant movement:** In the first scenario, the motor cover is placed on a two-meter conveyor belt, illustrated on the top row of the sequence of Figure 10. The conveyor belt moves the motor cover in a straight line at a constant speed of 50 mm/s and both direction and speed are not known by the robot beforehand. Through the experiments, the motor cover and robot are placed in similar initial positions, with slight variations in position.
- **Irregular movement:** In the second scenario, the motor cover is placed on top of a table trolley that is irregularly pushed by a human as depicted on the bottom row of the sequence of Figure 10, resulting in unpredictable and fluctuating movements. During the tests, subjects try to maintain a speed equivalent to the 50 mm/s present

in the conveyor belt. The combined effect of the human pushing and the instability of the trolley mechanism ultimately results in a movement that roughly corresponds to the target speed, yet with noticeable variations in velocity and acceleration.

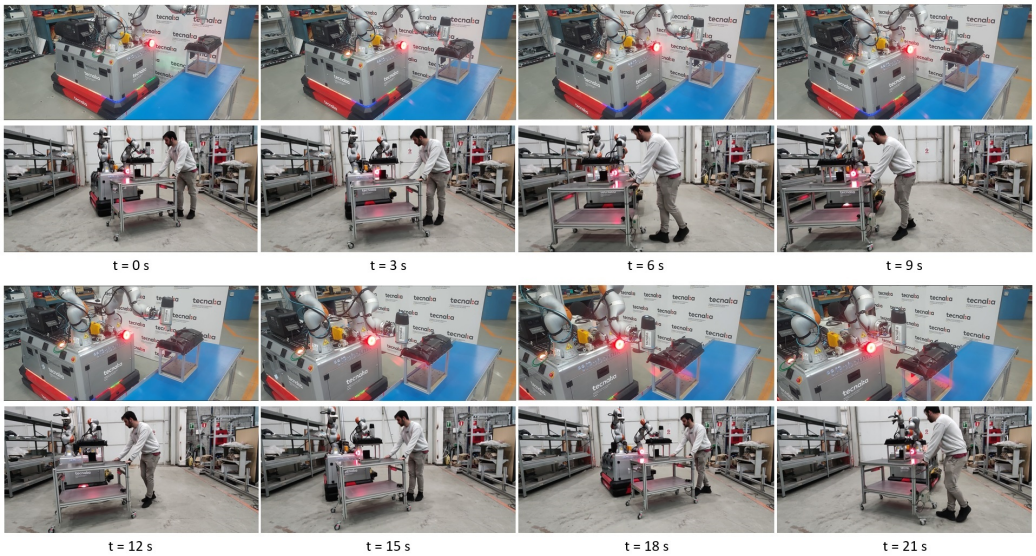

Figure 10. Motor cover on a conveyor belt (top row of the sequence) and moved manually on a trolley (bottom row of the sequence).

For each scenario, the screw-fastening sequence is repeated 10 times, resulting in 40 screws fastened in each scenario. The screw fastening is considered successful as long as the screw's bearing surface makes contact with the loading surface of the motor case. On the contrary, when some bolt threads are still visible, it is considered an unsuccessful screwing. The main failure reasons are the inability to reach the *gripper activation* state caused by large tool positioning errors in the previous *arm destination approach* state, as well as an unsuccessful coupling between the screwdriver and bolt head in the initial phase of the *gripper activation* state due to positioning errors during this insertion. Although the state machine can manage this state transition to *gripper activation* state, it is not able to handle the coupling error, as there is not any reliable feedback from the screwdriver that may trigger a new gripper activation attempt.

Initially, the time performance and success rate of the architecture are measured in both scenarios. Table 1 summarizes the obtained results. The first column indicates the movement type of the motor cover. The second and third columns provide information about the mean time spent in the complete screwing maneuver in each bolt, as well as the standard deviation. The fourth and fifth columns specify the mean duration and the standard deviation of the screwing phase where the robot activates the screwdriver. Finally, the last column details the success rate of the screw-fastening task for each movement type.

Table 1. Results obtained during the validation process.

	Time/Screw		**Screwing**		
	μ	σ	μ	σ	**Success**
Constant movement	9.45 s	2.07 s	5.28 s	1.88 s	92.5%
Irregular movement	11.55 s	2.51 s	7.15 s	2.59 s	67.5%

The results show a better performance of the system with a constant movement of the target, reaching a success rate of 92.5%. Specifically, with a constant movement of the target, the robot spends a mean time of 9.45 seconds per screw, including the approach and retract maneuvers, where around five of the seconds are spent in the screw-fastening

phase. In the case of irregular target movements, the success rate drops to 67.5%, spending around 2 s more in the process due to the difficulties of reaching the destination poses. In both scenarios, all the unsuccessful screwings are caused by an incorrect coupling between the screwdriver and bolt heads in the *gripper activation* state.

Delving into the error during the screwing phase, Table 2 shows the performance of the system while positioning the tool on the moving screw (*gripper activation* and *operation* states). The second and third columns indicate the minimum and maximum translation errors in positioning the tool on the target screwing pose, as well as the mean translation error on the fourth one. The subsequent three columns illustrate the minimum, maximum, and mean rotation errors of the screwing process.

Table 2. Errors measured during the screwing phase.

	Translation Error			**Rotation Error**		
	min	**max**	μ	**min**	**max**	μ
Constant movement	0.11 mm	8.83 mm	1.57 mm	0.006°	1.76°	0.097°
Irregular movement	0.11 mm	20.5 mm	3.82 mm	0.006°	2.91°	0.228°

The obtained error measurements show the performance difference between both movement types, especially in translation errors. During constant movement, the mean translation error is around 1.57 mm, enough to ensure correct insertion of the screwing tool in the bolts. In the case of irregular target movement, the mean error rises to 3.8 mm, which causes an unsuccessful contact between the tool and bolt, which significantly reduces the success rate. The abrupt velocity and acceleration changes on the irregular movement are highlighted in the maximum translation error, where the value grows up to 20 mm.

Figure 11 further illustrates the difference in positioning error of the arm throughout constant versus irregular motion. The smooth error curve of the constant motion setup (blue line) points out stable tracking and efficient phase transitions. In contrast, fluctuations in motion are manifested in the irregular motion setup (orange line), resulting in more gradual error reduction, later phase transitions, and noticeable error peaks during the last phases of the operation. These factors, as well as the reduced rate of success seen in the irregular motion case, reflect the challenge that erratic motion entails for an effective screwing operation when the target is moving irregularly.

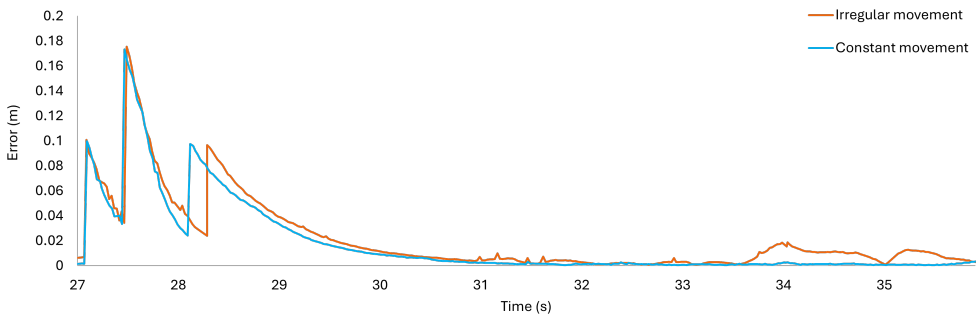

Figure 11. Arm positioning error for the robotic arm, while tracking the object that followed either constant (blue) or irregular movement (orange).

To sum up, there are two main conclusions extracted from the experiments carried out to validate the presented architecture. On the one hand, the mobile visual servoing algorithm can track and follow the motor cover, reaching an accuracy level sufficient to perform the screw fastening with a constant motion pattern. On the other hand, irregular movements increase significantly the tool positioning error, causing a substantial fall in the success rate; an abrupt error rise during the activation of compliance, where the robot tries to reach contact with the bolt, causes a large momentary deviation that finishes with the tool

on a side of the bolt head without any option to recover from the error. Nevertheless, the system shows good performance, being able to carry out a demanding task like screwing in moving parts.

10. Conclusions and Future Work

This paper builds on solid and well-established theoretical, technological, and methodological foundations to develop a practical application of mobile manipulators for precise industrial tasks involving motion. The contribution of this work is focused on its practical implications for the development of industrial automated solutions for the assembly of moving parts. The paper addresses relevant topics like task precision and stability using mobile manipulators in dynamic environments, including features like compliance or the use of state machines to guide the different steps of the operation. The practical utility of this research has been validated through a screw-fastening operation conducted on a moving assembly line.

Specifically, the presented paper proposes an architecture for mobile visual servoing able to control both a mobile platform and a manipulator based on visual feedback, addressing specific problems of performing operations on moving parts in industrial environments. The presented work proposes a state machine-driven control architecture able to guide the complete industrial task by modifying the different parameters of the control modules. Additionally, the architecture includes a compliance controller that complements the control law to ensure safe contact between the tool and part for a successful operation.

The designed approach was implemented on a mobile manipulator composed of a holonomic mobile platform and a Kuka LBR iiwa arm, equipped with an automatic screwdriver and a 2D industrial vision system. This implementation was tested on a screw-fastening operation on a moving engine cover, a demanding task due to the accuracy required for its successful completion. The testing involved the bolt screwing while the engine was transported on a conveyor belt, a traditional scenario where the part moves linearly, as well as when it was transported manually on a trolley with random movements. The architecture achieved a 92.5% success rate with a positioning error of 1.5 mm during the tests on the conveyor belt, although the performance fell significantly in the tests involving irregular movement, with a success rate of 67.5% and a positioning error of 3.8 mm. Even so, the system showed promising performance in a demanding operation like the screw fastening on moving parts.

As future steps, there are several open issues to address. Initially, the system was tested using markers to simplify the target pose detection. The marker removal would be a huge step towards the application of this kind of solution in industry, although the detection and tracking precision could be an issue for the performance of the approach. Additionally, the state machine is focused on operations involving the placement and activation of the tool on the desired positions of the moving part, which covers a great amount of industrial tasks. Even so, polishing or spraying applications follow a different strategy and require other input data. Therefore, it would be interesting to test the architecture with multiple state machines to validate it in various industrial applications, enhancing the impact of the proposed approach. Finally, the decline in performance associated with irregular motion is a critical issue that may be caused by several factors, such as latency, quality issues in perception, or limitations in the control system's responsiveness and predictive capabilities. Besides the improvements in the perception systems to enhance image quality and pose estimation, transitioning to more advanced control algorithms beyond the PID algorithm can address the lack of responsiveness to irregular motion. Options include optimal or adaptive control, or more sophisticated methods like model predictive control (MPC).

Author Contributions: Conceptualization, J.G.H. and A.I.; methodology, J.G.H. and A.I.; software, J.G.H. and A.I.; validation, J.G.H. and A.I.; formal analysis, J.G.H.; investigation, J.G.H. and A.I.; resources, J.G.H.; data curation, A.I.; writing—original draft preparation, A.I.; writing—review and editing, J.G.H. and A.I.; visualization, J.G.H.; supervision, A.I. All authors have read and agreed to the published version of the manuscript.

Funding: This work has received funding from the European Union Horizon 2020 research and innovation program as part of the project ODIN under grant agreement No. 101017141.

Data Availability Statement: The experiment data are available at https://doi.org/10.5281/zenodo.10820175 (accessed on 1 March 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Hanafusa, H.; Yoshikawa, T.; Nakamura, Y. Analysis and control of articulated robot arms with redundancy. *IFAC Proc. Vol.* **1981**, *14*, 1927–1932. [CrossRef]
2. Miller, R.K. How an Industrial Robot Works. In *Industrial Robot Handbook*; Springer: Boston, MA, USA, 1989; pp. 10–25.
3. Maxwell, W.L.; Muckstadt, J.A. Design of automatic guided vehicle systems. *Iie Trans.* **1982**, *14*, 114–124. [CrossRef]
4. Holcombe, W.D.; Dickerson, S.L.; Larsen, J.W.; Bohlander, R.A. Advances in guidance systems for industrial automated guided vehicles. In *Proceedings of the Mobile Robots III*; SPIE: Bellingham, WA, USA, 1989; Volume 1007, pp. 288–297.
5. Marvel, J.; Bostelman, R. Towards mobile manipulator safety standards. In Proceedings of the 2013 IEEE International Symposium on Robotic and Sensors Environments (ROSE), Washington, DC, USA, 21–23 October 2013; pp. 31–36. [CrossRef]
6. Markis, A.; Papa, M.; Kaselautzke, D.; Rathmair, M.; Sattinger, V.; Brandstötter, M. Safety of Mobile Robot Systems in Industrial Applications. In Proceedings of the ARW & OAGM Workshop, Steyr, Austria, 9–10 May 2019. [CrossRef]
7. Sanderson, A.C.; Weiss, L.E. Adaptive visual servo control of robots. In *Robot Vision*; Springer: Berlin/Heidelberg, Germany, 1983; pp. 107–116.
8. Weiss, L.; Sanderson, A.; Neuman, C. Dynamic sensor-based control of robots with visual feedback. *IEEE J. Robot. Autom.* **1987**, *3*, 404–417. [CrossRef]
9. Espiau, B.; Chaumette, F.; Rives, P. A new approach to visual servoing in robotics. *IEEE Trans. Robot. Autom.* **1992**, *8*, 313–326. [CrossRef]
10. Lippiello, V.; Siciliano, B.; Villani, L. Position-based visual servoing in industrial multirobot cells using a hybrid camera configuration. *IEEE Trans. Robot.* **2007**, *23*, 73–86. [CrossRef]
11. Castelli, F.; Michieletto, S.; Ghidoni, S.; Pagello, E. A machine learning-based visual servoing approach for fast robot control in industrial setting. *Int. J. Adv. Robot. Syst.* **2017**, *14*, 1729881417738884. [CrossRef]
12. Wu, H.; Andersen, T.T.; Andersen, N.A.; Ravn, O. Application of visual servoing for grasping and placing operation in slaughterhouse. In Proceedings of the 2017 3rd International Conference on Control, Automation and Robotics (ICCAR), Nagoya, Japan, 24–26 April 2017; pp. 457–462.
13. Pitipong, S.; Pornjit, P.; Watcharin, P. An automated four-DOF robot screw fastening using visual servo. In Proceedings of the 2010 IEEE/SICE International Symposium on System Integration, Sendai, Japan, 21–22 December 2010; pp. 379–383.
14. Hao, T.; Xu, D. Robotic grasping and assembly of screws based on visual servoing using point features. *Int. J. Adv. Manuf. Technol.* **2023**, *129*, 3979–3991. [CrossRef]
15. Ruybal, K.R.; Lumia, R.; Wood, J.E. A finite state machine approach to visual servoing to increase positional accuracy of impedance controlled robots. *Int. J. Recent Adv. Mech. Eng. (IJMECH)* **2013**, *2*, 1–14.
16. Aref, M.M.; Ghabcheloo, R.; Kolu, A.; Hyvönen, M.; Huhtala, K.; Mattila, J. Position-based visual servoing for pallet picking by an articulated-frame-steering hydraulic mobile machine. In Proceedings of the 2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM), Manila, Philippines, 12–15 November 2013; pp. 218–224.
17. Fue, K.; Porter, W.; Barnes, E.; Li, C.; Rains, G. Center-articulated hydrostatic cotton harvesting rover using visual-servoing control and a finite state machine. *Electronics* **2020**, *9*, 1226. [CrossRef]
18. Wang, Y.; Lang, H.; De Silva, C.W. A hybrid visual servo controller for robust grasping by wheeled mobile robots. *IEEE/ASME Trans. Mechatronics* **2009**, *15*, 757–769. [CrossRef]
19. Belmonte, Á.; Ramón, J.L.; Pomares, J.; Garcia, G.J.; Jara, C.A. Optimal image-based guidance of mobile manipulators using direct visual servoing. *Electronics* **2019**, *8*, 374. [CrossRef]
20. Li, W.; Xiong, R. A hybrid visual servo control method for simultaneously controlling a nonholonomic mobile and a manipulator. *Front. Inf. Technol. Electron. Eng.* **2021**, *22*, 141–154. [CrossRef]
21. Arora, P.; Papachristos, C. Mobile manipulator robot visual servoing and guidance for dynamic target grasping. In Proceedings of the Advances in Visual Computing: 15th International Symposium, ISVC 2020, San Diego, CA, USA, 5–7 October 2020; Proceedings, Part II 15; Springer: Berlin/Heidelberg, Germany, 2020; pp. 223–235.
22. Burgess-Limerick, B.; Lehnert, C.; Leitner, J.; Corke, P. An Architecture for Reactive Mobile Manipulation On-The-Move. In Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA), London, UK, 29 May–2 June 2023; pp. 1623–1629. [CrossRef]
23. Taubin, G. 3D Rotations. *IEEE Comput. Graph. Appl.* **2011**, *31*, 84–89. [CrossRef] [PubMed]
24. Knospe, C. PID control. *IEEE Control. Syst. Mag.* **2006**, *26*, 30–31. [CrossRef]
25. Davidson, J.K.; Hunt, K.H.; Pennock, G.R. Robots and screw theory: Applications of kinematics and statics to robotics. *J. Mech. Des.* **2004**, *126*, 763–764. [CrossRef]
26. Lee, C.G. Robot arm kinematics, dynamics, and control. *Computer* **1982**, *15*, 62–80. [CrossRef]

27. Hogan, N. Impedance control: An approach to manipulation. In Proceedings of the 1984 American Control Conference, San Diego, CA, USA, 6–8 June 1984; pp. 304–313.
28. Hogan, N. Impedance control: An approach to manipulation: Part II—Implementation. In Proceedings of the 1984 American Control Conference, San Diego, CA, USA, 6–8 June 1984.
29. Mitchell, W.C.; Staniforth, A.; Scott, I. *Analysis of Ackermann Steering Geometry*; Technical Report, SAE Technical Paper; SAE International: Warrendale, PA, USA, 2006.
30. Taheri, H.; Zhao, C.X. Omnidirectional mobile robots, mechanisms and navigation approaches. *Mech. Mach. Theory* **2020**, *153*, 103958. [CrossRef]
31. Gfrerrer, A. Geometry and kinematics of the Mecanum wheel. *Comput. Aided Geom. Des.* **2008**, *25*, 784–791. [CrossRef]
32. Kuka LBR iiwa. Available online: https://www.kuka.com/en-us/products/robotics-systems/industrial-robots/lbr-iiwa (accessed on 1 March 2024).
33. IDS. Available online: https://en.ids-imaging.com (accessed on 1 March 2024)..
34. OnRobot. Available online: https://onrobot.com/en/products/onrobot-screwdriver (accessed on 1 March 2024).
35. ros. Available online: https://www.ros.org (accessed on 1 March 2024).
36. Aruco ROS. Available online: https://wiki.ros.org/aruco_ros (accessed on 1 March 2024).