Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN

INGEGNERIA BIOMEDICA, ELETTRICA E DEI SISTEMI

Ciclo 33

Settore Concorsuale: 09/G1 - AUTOMATICA

Settore Scientifico Disciplinare: ING-INF/04 - AUTOMATICA

ROBOTIC SENSING AND MANIPULATION OF DEFORMABLE LINEAR OBJECTS WITH LEARNING-BASED METHODS

Presentata da: Riccardo Zanella

Coordinatore Dottorato

Michele Monaci

Supervisore

Claudio Melchiorri

Co-supervisore

Gianluca Palli

Esame finale anno 2021

ALMA MATER STUDIORUM UNIVERSITY OF BOLOGNA

Abstract

School of Engineering and Architecture
Department of Electronics, Computer Sciences and Systems

Doctor of Philosophy in Biomedical, Electrical and System Engineering

Robotic Sensing and Manipulation of Deformable Linear Objects with Learning-based methods

by Riccardo Zanella

Nowadays robotic applications are widespread and most of the manipulation tasks are efficiently solved. However, Deformable Objects (DOs) still represent a huge limitation for robots. The main difficulty in DOs manipulation is dealing with the shape and dynamics uncertainties, which prevents the use of model-based approaches (since they are excessively computationally complex) and makes sensory data difficult to interpret.

This thesis reports the research activities aimed to address some applications in robotic manipulation and sensing of Deformable Linear Objects (DLOs), with particular focus to electric wires. In all the works, a significant effort was made in the study of an effective strategy for analyzing sensory signals with various machine learning and deep learning algorithms.

In the former part of the document, the main focus concerns the wire terminals, i.e. detection, pose estimation, grasping, and insertion. First, a pipeline that integrates vision and tactile sensing is developed. Then, further improvements are proposed for each module in subsequent works. A novel procedure is proposed to gather and label massive amounts of training images for object detection in the field with minimal human intervention. Together with this strategy, we extend a generic object detector based on Convolutional Neural Networks (CNNs) for orientation prediction. The insertion task is initially addressed in open-loop by grasping the wire close to the terminal, while a regressor fed with tactile sensor data is used to analyze the contact forces and verify the success. Then, also this component is extended by developing a closed-loop control capable to guide the insertion of a longer and curved segment of wire through a hole, where the contact forces are estimated by means of a Recurrent Neural Network (RNN). The approach here proposed shows how a cheap sensor embedded with suitable artificial intelligence can provide information comparable to a more expensive force sensor.

In the latter part of the thesis, the interest shifts to the entire DLO and to its shape. Robotic reshaping of a DLO is addressed by means of a sequence of pick-and-place primitives driven by visual data. In the proposed system, a decision making process learns the optimal grasping locations exploiting Deep Q-learning and finds the best releasing point. The success of the solution leverages on a reliable interpretation of the DLO shape. For this reason, further developments are made on the visual segmentation.

Contents

Chapter 1

Introduction

Deformable Object (DO) is an umbrella term that covers all the objects capable of changing their shape under the action of a force. DOs are very common in our everyday life and are present in every setting, from natural to human environment and from houses to factories. Few example objects that can be considered deformable are cloths, pillows, curtains, electric cables, paper sheets, food products, plants, vegetables, human/animal bodies. A common classification of DOs is based on their geometry, and it defines three classes:

- uniparametric or linear objects, commonly known as Deformable Linear Objects (DLOs), are all those DOs having one dimension significantly larger than the other two (e.g. cables, strings, ropes and wires);
- biparametric or planar objects, are those having one dimension considerably smaller than the other two (e.g. paper, cloths and leaves);
- triparametric or volumetric objects are those like sponges, plush toys, bread and flesh.

Another useful categorization for DOs uses their physical properties. In this way they can be 1) plastic, if they hold the undergone deformation in response to applied forces also after these are removed; 2) elastic, if they return to the original shape and size after the forces that induce the deformation are removed. However, most of the DOs have elasto-plastic properties, which means that they exhibit elastic behavior for small loads, but when the load is further increased the material can undergo plastic deformation. Electric wires, paper and bread are examples of this last category.

As already stated, DOs are present in every domain, thus DOs manipulation is an essential skill for robots to enter the human living and working environments or to extend their involvement in industrial applications. For instance, robots could be able to handle fragile products in the food industry and in agriculture; they could become more involved in caregiving activities (e.g. dressing, feeding) for the elderly and disabled; they could be employed medical or surgery applications (e.g. manipulating catheter, electrode arrays, surgery threads); or, they could manipulate flexible objects to lessen physical burden on workers in manufacturing plants.

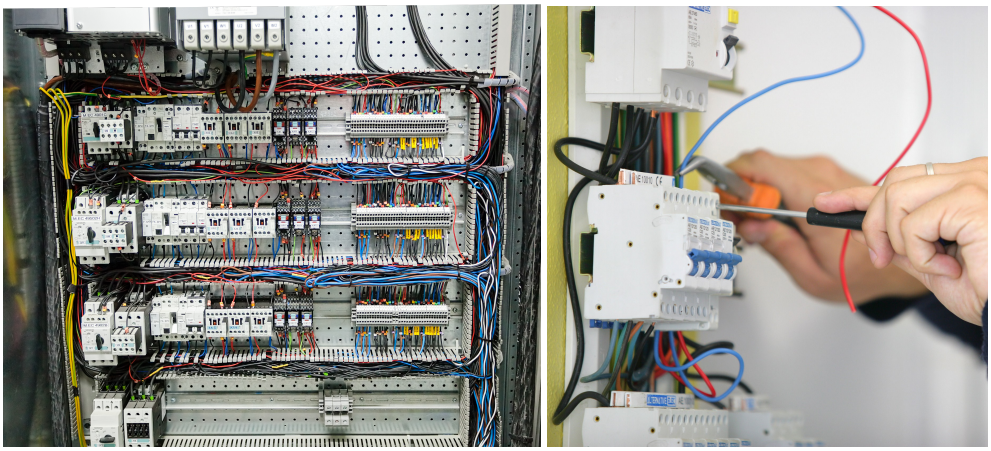
FIGURE 1.1: Examples of deformable objects.

Unfortunately, robotic manipulation of DOs can be extremely difficult due to the infinite amount of possible state configurations. Manipulation of rigid objects consists basically in changing their pose (position and orientation), while avoiding collisions, in the context of pick-and-place or assembly tasks. When DOs are involved, instead, manipulation in most cases also affects their shape, with geometrical or topological changes. Unlike rigid objects, whose dynamics are well-understood, the motion of a DO depends on a large and complex set of parameters that define its stiffness, friction, and volume preservation.

Therefore, recognize the DO state is a crucial issue in robotic manipulation. Sensing devices, like cameras, tactile or force/torque sensors, are employed to perceive the deformations and the shape. A dynamics model or a physics simulator of the DOs can also be built by estimating the physical parameters from the sensors. Similarly to the manipulation of rigid objects, a dynamics model could be used to design a suitable control system [1, 2]. Unfortunately, the complicated dynamics would result in a significant difficulty in modeling, or in an unrealistic and useless model. Alternatively, physics simulations can be incorporated into the motion planner to generate the robot trajectory based on the simulated deformation [3, 4]. However, simulators are difficult to calibrate to be consistent with real-world physics. In particular, the quality of the simulation is extremely sensitive to the model parameters. In addition, running a physically-based simulation is time-consuming and thus infeasible for estimating fast or large deformation in real time. Due to these difficulties, it is often preferable to find a robot trajectory or a control action that relies solely on sensor feedbacks [5, 6, 7, 8, 9] or possibly to encode the sensor data on a geometric description of the object state [10]. Even this approach is not straightforward, since sensor data concerning DOs may be hard to interpret and use effectively for extracting the information sought. This difficulty can be circumvented by utilizing deep learning-based approaches [8, 11, 12, 13], at the price of having to create, if not existing, large training dataset [14] (raging among the different shapes configuration, points of view, light setups, etc.). Moreover, in many realistic applications, operating conditions on the task or constructive constraints on the setup may lead to limitations in the sensory systems that prevent the use of some approaches. For example, when significant occlusions are present, an integration of vision data with forces or tactile feedback should be considered [7]. On the other hand, there are also complex problems that require a combination of multiple feedbacks to reduce the information uncertainty or acquire a certain level of perception [5].

This thesis focuses on DLOs, with particular interest in electric wires. This specific category of DOs is commonly used in an extremely wide set of applications, ranging from Information and Communications Technology to constructions and from industrial manufacturing to power distribution systems.

For instance, in the manufacturing industry, *switchgear wiring* represents a very challenging industrial application from the robotic manipulation point of view. Switchgears and control panels (shown in Figure 1.2) are basic components of power generation and distribution stations, commercial and institutional buildings, industrial plants and automated factories, automatic machines and civil houses. In the actual scenario, the switchgear wiring is mostly executed by human operators because of the complex manipulation tasks and the large variability of the design, usually characterized by highly-customized solutions and small lot or single item production. In general, the wiring is composed of a sequence of single wire connections. Each wire connection implies: 1) the localization in the switchgear of the components to be connected, i.e. the points in which the two wire ends will be placed



(A) Example of a switchgear.

(B) Connection of a wire in a electromechanical component.

FIGURE 1.2: Switchgear wiring.

(since their position is known with some uncertainty); 2) first wire-terminal connection; 3) wire routing inside the wire collector; 4) second wire-terminal connection. Steps from 2) to 4) must be repeated for all the connections in the switchgear net list. Moreover, step 2) can be further decomposed in the following phases: i) wire localization; ii) wire grasping; iii) wire pose detection and correction; iv) insertion into the terminal; v) tightening of the terminal screw (that can be achieved by a screwdriver or is automatic depending on the terminal type); vi) wire connection check. Even a single wire connection is really a challenging task for a robotic system, since it can be seen as a sub-millimeter precision peg-in-hole problem involving the manipulation of a deformable object, i.e. the wire. The terminals are also difficult to see and to access, due to their location on both sides of the electromechanical components, the proximity of other components and wire collectors, and the presence of previously connected wires. These issues limit the applicability of vision systems to guide and control the wiring process.

Wiring harness assembly in cars is another very complicated manipulation activity executed entirely by hand in the automobile industries along the production chain. In this example, the wiring harness is initially in an unknown configuration and needs to be untangled and arranged in the body of the car, plugging the connectors to all the electromechanical components. This requires to reshape the cables, make them pass through narrow holes and fix them to the specific supports on the body of the car. Figure 1.3 reveals the incredibly complex wiring harness of a Bentley's car.

Generally speaking, in a robotic application that involves DLOs manipulation, we can identify some *recurring and basic tasks*, which can be also divided into subtasks or smaller challenges. Every robotic application usually starts by **grasping** the object to handle. To perform this task the robot needs to know where the object located is in the space, with respect to its reference frame. This information often comes from a vision system, which detects the object and estimates its pose [14]. In several applications with DLOs, the system also requires a shape estimation and a suitable representation [10, 15, 16] of the DLO. This information is particularly necessary when the specific grasping point matters to the task. In fact, there are infinite points that are eligible candidates for grasping along a DLO and they are all closely coupled, hence moving one of them or another entail different deformation and state changes. After grasping, other two very common and basic tasks in DLO

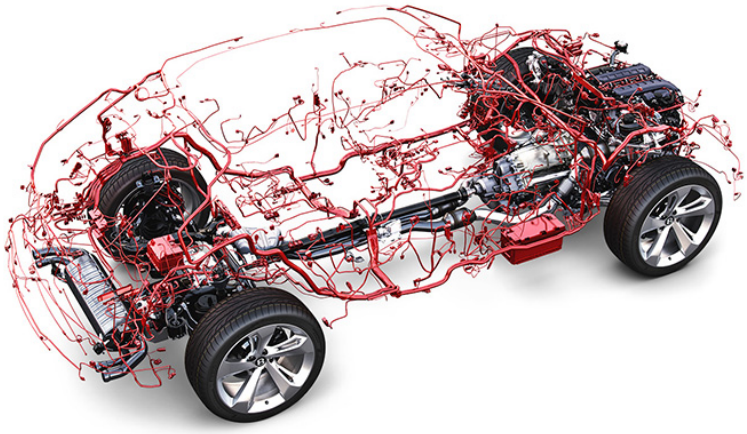


FIGURE 1.3: Wiring harness scheme of a Bentley Bentayga SUV.

manipulation are the **reshaping** and the **insertion**. In both the cases dynamic model, physics simulator and shape representation can be employed. The shape control often requires visual feedback for planning the robot trajectory in order to minimize the errors between the current and the target shape [16]. Whereas, tactile or force/-torque data can be exploited, especially for inserting, to predict the deformation by means of a model [17] or they can directly be used to drive the manipulator [18, 8, 19]. A common approach in most of the works published in the last decade is the use of deep-learning methods, in particular to extract useful information from visual data. For example, some authors proposed to employ a CNN for learning an inverse model of the DLO by freely interacting with it and then use this model for imitating human demonstrations [20, 21]. In other works the authors proposed to learn instead a forward model of the DLO dynamics, in a state or latent space, and use it for planning the robot actions with a model predictive control [22, 16]. Recent success in deep reinforcement learning also provides a promising direction for learning to manipulate DOs with data driven methods by directly mapping the input images into the robot actions [23, 24, 25, 26].

The works presented in this thesis cover all the aforementioned *basic tasks* (i.e. grasping, reshaping and inserting), proposing in each case a data-driven solution that embeds suitable learning-based algorithms. Particular attention is paid to the deformation sensing and shape representation, exploiting vision and tactile sensors.

The research activities reported in this thesis are motivated and supported by *WIRES (Wiring Robotic System for Switchgears)* an ECHORD++ experiment and *REMODEL (Robotic technologies for the manipulation of complex deformable linear objects)*, a four-year project funded by the European Commission in the Horizon 2020 programme. WIRES aimed to develop tools and techniques for enabling the robotized switchgear wiring in the industrial scenario. Whereas, REMODEL plans to extend the results of the ECHORD++ WIRES, by enabling the implementation of manufacturing activities involving the manipulation of complex deformable objects in industrial automated production lines. The main target is the robotic handling of DLOs, by deeply understanding their complex behavior, including elasticity and plasticity, and embedding this knowledge, together with proper manipulation skills and perception, based on vision and tactile sensing, into a bimanual robotic system.

The remaining of this document can be organized as follow:

The **first part** focuses on the insertion of a DLO in a hole.

- In chapter 2, a pipeline that integrates vision and tactile feedback for grasping and inserting an electrical wire in a hole is presented.
- In chapter 3, the vision modules of the aforementioned pipeline are extended and a new strategy for 3DoF pose estimation is proposed.
- In chapter 4, also the task of inserting a wire inside a hole is improved by exploiting a closed-loop control.

The **second part** focuses on the reshaping of a DLO.

- In chapter 5, an autonomous system for reshaping a soft DLO from visual data is presented.
- In chapter 6, two state-of-the-art algorithms for image segmentation are trained using an auto-generated dataset of electric wires to improve the visual perception.

Chapter 2

A Robotic System for Wire-Terminal Pick & Insertion Tasks

The electric wiring task, as already outlined in chapter 1 for the use case of a switchgear, consists of a sequence of single wire connections that generally starts and ends with the insertion of the wire terminal into a hole. In this chapter we investigate this problem in details by braking it down in subtasks, i.e. detect the terminal, estimate its planar pose for the grasping, verify the grasp, estimate the new terminal pose in the gripper, perform the insertion while verifying the correctness.

2.1 Introduction

In this work, we present how to complementary exploit and combine vision and tactile feedback with several machine learning approaches to solve the problem of wire detection, manipulation and insertion into the terminal hole. It is worth mentioning that, even if no occlusion given by other wires or components is present in the experimental setup, the gripper used to grasp the wire generally prevents by itself the usage of a vision system to provide a close view of the terminal from front. It will be shown that the features and characteristics of the vision and tactile sensors are complementary for the task at hand. As a matter of fact, they both provide information that are strictly needed to achieve the required precision. This is due by the fact that, in a realistic scenario, the tight spaces between wires and components prevent the use of the vision system, in particular during the final part of the wire insertion, also when the same task is performed by a human being. It results that, in most of the practical cases, when a sensor can generate useful information, the other can not and vice versa. In the insertion phase, the tactile sensor can be used to quantify a collision, in order to evaluate if the insertion is correctly achieved. Moreover, in this work a novel technique to automatize and speed up the generation of training datasets is presented. This technique is exploited to train a Convolutional Neural Network (CNN) in order to detect small objects (like wire terminals), and a new method to estimate their 3D positions using multiple CNN predictions is shown. Additionally, Multi-Layer Perceptrons (MLPs), Random Forests (RFs) and Support Vector Machines (SVMs) are either trained to produce an affordable output from the tactile sensor to evaluate the correctness of the wire insertion task and detect faults. The research paper [7] which introduced the results presented in this chapter has been awarded as *2020 IEEE T-ASE Best New Application Paper*. We may additionally affirm that the application considered in this work can be of great interest for the overall robotic community since it implies several issues related to micro-assembly

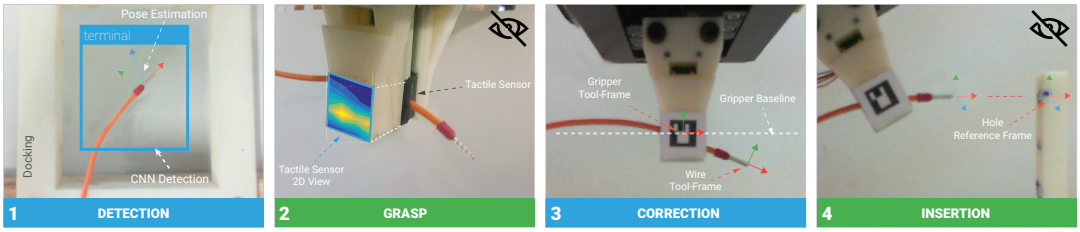


FIGURE 2.1: The system's pipeline is composed by four components: 1) The wire detection, that exploits a self-trained multiple CNN triangulation for pose estimation; 2) The wire grasp during which a tactile sensor is used together with MLP to classify the grasp; 3) The pose correction of the wire; 4) The wire insertion into the terminal hole exploits a the tactile sensor and a MLP trained to detect collisions.

processes, manipulation of deformable objects, occlusion in computer vision systems, tactile sensing, vision-tactile fusion and machine learning.

The chapter is organized as follows. section 2.2 reports a summary of the previous researches carried out by other authors in this field. section 2.3 introduces an high-level vision of the wire insertion problem. section 2.4 presents the hardware setup exploited during experiments. section 2.5 describes in detail the components of the system pipeline. Finally, section 2.6 reports the set of experimental tasks that show the effectiveness of the proposed approach.

2.2 Related Works

A number of previous research activities can be found in literature about the modeling, the manipulation and the visual tracking of Deformable Linear Objects (DLOs) such as electric wires, demonstrating the large interest in this field.

In [27] a method to calculate the force acting on a purely-elastic flexible wire from its shape observed by stereo vision is developed. The same authors presented a method to insert a purely-elastic flexible wire into a hole observing the shape of the wire by stereo vision in [28]. The task of picking up cables from approximately known positions with an industrial robot using two light barriers has also been investigated in [29]. In [30, 31] the authors presented the static and the dynamic modeling of DLOs based on differential geometry coordinates, respectively. In [32] a path planning algorithm for DLOs subject to manipulation constraints is presented. In [33] a motion planner for manipulating DLOs and tying knots using two cooperating robotic arms is developed. In [34] a DLOs model based on mechanically rigorous and geometrically exact dynamic splines including both elastic and plastic deformation is described. In [35] a modeling of electric cables based on the visual measurement of their static and dynamic deformation is performed for cable insertion in electric and automotive industries. In [36] an algorithm for tracking DLOs based on a probabilistic generative model that incorporates observation and the physical properties of the tracked object is presented. In [3] the manipulation planning problem of a DLO handled by a gripper at one of its extremities in free or contact space is considered. In [37], the robot is guided to grasp the wire on the clamp cover adopting a SIFT (Scale-Invariant Feature Transform) based algorithm. The problem of assembling flexible wire harness into instrument panel frame is addressed in [38] by making use of vision sensors and markers attached on the surfaces of the clamp.

Tactile sensing and vision are two synergistic modalities for manipulation. Vision systems provide rich information regarding unknown objects, in fact they became one of the main feedback source in robotics. However, they are often difficult to be applicable when the objects are occluded or visually confused. Recent progress in artificial touch sensing hardware allowed the robotics community to endow robots with touch capabilities and to show that tactile sensing can be efficiently employed in robot grasping. In order to deal with complex tactile information, machine learning algorithms have been widely used to address the classification problem. A grasp detection deep network is proposed in [39] to detect the grasp rectangle from the visual image with a new metric to assess the stability of the grasp. In [40] a novel method to systematically solve the visual-tactile fusion in object recognition tasks using multivariate time series is developed. In [41] visual modality is used to aid learning tactile modality during the training phase. In [42] the authors propose a cross-modal approach based on the use of visuo-tactile data for object recognition. A comparative analysis of classification algorithms for tactile sensors mounted on humanoid hand is presented in [43]. In [44] they present a robotic agent that learns to derive object grasp stability from touch. Classification is conducted through kernel logistic regression, applied to a low- dimensional approximation of the tactile data read from the robot's hand. The implementation of tactile object identification and feature extraction techniques is discussed in [45], where two methods of tactile data interpretation are combined on data acquired during a single unplanned grasp: a random forests classifier and parametric object property estimators.

In this work, our aim is to combine vision, tactile sensing and machine learning to manipulate electric wires and insert them into electromechanical components, taking into account the real manufacturing application constraints. This work will be part of an automatic switchgear wiring system under development. Previous approaches to similar manufacturing problems are mainly based on vision. The strength of the approach presented in this chapter relies mainly on the synergistic combination of vision and tactile data to overcome the application constraints. The way how to combine these sensors has been selected taking into account that there are working conditions in which one of these two sensors can be ineffective or unreliable. Moreover, the whole development is performed trying to reduce complexity and cost of the final system.

This manufacturing application is really relevant in the industrial scenario, since the switchgear wiring still today represents a completely manual operation, that in turn results the major cost in the production of these highly customized items. Moreover, the interest in this field is confirmed by the number of literature works and companies involved. The proposed work is relevant because first there is no literature dealing with the overall task sequence, secondly it investigates a novel solution based on the combination of different technologies, i.e. vision, tactile sensing and machine learning to solve a problem that remains unsolved in the actual industrial practice.

The assumptions considered in this work are the following:

- only a limited part of the complete wiring task is taken into account to simplify the analysis;
- since the system is designed for an industrial setting, we suppose to operate in a partially structured environment;
- the resolution of the vision system and tactile sensor is somehow limited by the use of low-cost devices.

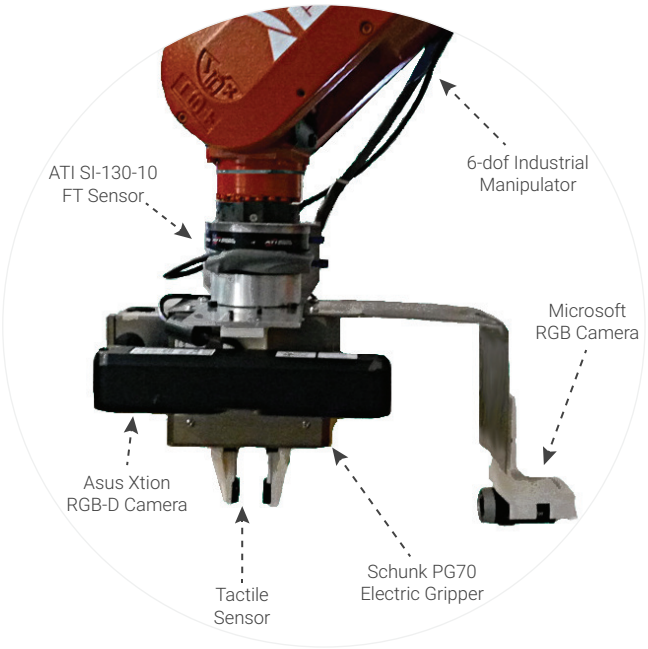


FIGURE 2.2: The end effector used during the experiments. The Microsoft RGB Camera (referred as *side Camera*) is mounted on the end effector for evaluation purposes. In the real scenario the *side Camera* will be fixed to ground, to reduce the end-effector encumbrance.

Besides this latter point is a benefit form the point of view of the overall system cost, resolution problems are mitigated by the introduction on suitable techniques, such as the exploitation of multiple views and machine learning, to achieve the desired task success rate.

2.3 Task Description

The task considered in this work consists in the insertion of an electric wire terminal in a hole that emulates the electromechanical component connector. Since it is really difficult to evaluate the correct alignment and the contact of the wire terminal using a real electromechanical component, an emulation body composed by a beam with a 5 mm pass-through hole has been used to ease the evaluation of the system performance without affecting results. With reference to Figure 2.1, the task to be executed by the robot is composed by the following operations: 1) Detect the pose of the wire terminal using vision feedback in order to grasp it; 2) grasp the wire and validate the grasp through tactile feedback to evaluate if the following steps can be correctly executed; 3) Estimate the pose of the wire end w.r.t. the gripper with the required precision using vision feedback in order to correctly execute the insertion task; 4) Execute the wire insertion into the terminal hole detecting possible collisions by means of tactile feedback.

In the 2nd frame of Figure 2.1 the *blind icon* is shown to emphasize that the vision system can't be used for monitoring the scene in this phase since the grasp, obviously, needs to be monitored by a device able to detect the physical contact with the object, e.g. the tactile sensor. The same holds for the 4th frame, showing the wire insertion, since in the real scenario the limited space available and the presence of other components and wires in the neighborhood of the working region prevent the scene observation by the camera, because both of occlusion problems and the

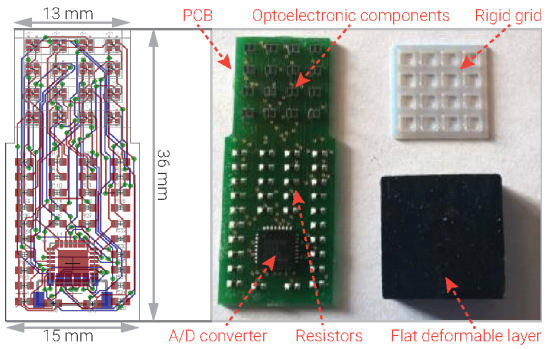


FIGURE 2.3: Layout of the tactile sensor PCB on the left and pictures of the sensor components on the right.

impossibility of placing the camera close to the fingers. It results that the designed tactile sensor only fits with the available space in the region close to the wire connection point in the application scenario. This assumption is clearly not true in our experimental setup created ad-hoc to evaluate the effectiveness of the system, but we selected to not use the vision during the insertion to recreate the real working conditions.

2.4 The Experimental Setup

The hardware setup used during the experiments here described is shown in Figure 2.2. The system is composed by an industrial manipulator, a COMAU Smart Six, equipped with a commercial gripper, a Schunk PG70 electric parallel gripper and an ATI SI-130-10 Force Torque (FT) sensor mounted on the wrist (between the robot interface and the gripper). Moreover, an Asus Xtion 3D camera with VGA resolution is mounted on the one gripper side pointing downward, to provide a top view of the scene (namely, the *hand camera* in the following). The 3D feature of this camera is useful for the reconstruction of the component location and encumbrance in the switchgear (these problems are not treated in this work), but the 3D resolution is too poor for the purpose of wire terminal detection and grasping. Therefore, a computer vision algorithm has been developed, as reported in subsection 2.5.2, for reconstructing the 3D pose of the wire using multiple RGB images only provided by this camera.

On the other hand, to provide a close view of the task execution, an additional Microsoft 2D LifeCam camera with HD resolution is mounted on one gripper side (namely, the *side camera* in the following). In the experiments here reported, this camera was used also to estimate the wire pose after the grasp, as detailed in subsection 2.5.4. In normal conditions, this operation is performed by a fixed camera placed in a known position reachable by the robot to reduce the end-effector encumbrance.

A custom tactile sensor [46, 47] has been developed on the base of the one presented in [48] for the task here considered, see Figure 2.3, and it has been mounted on one gripper's finger to provide a tactile image of the grasped objects, e.g. the wire. It is constituted by 16 taxels organized as a 4×4 matrix and a deformable layer with a flat shape. Each taxel is constituted by a single SMT photo-reflector integrating both an infrared LED and a PhotoTransistor (PT). When a contact with the deformable layer occurs, it produces vertical displacements of the reflective surfaces of the cells

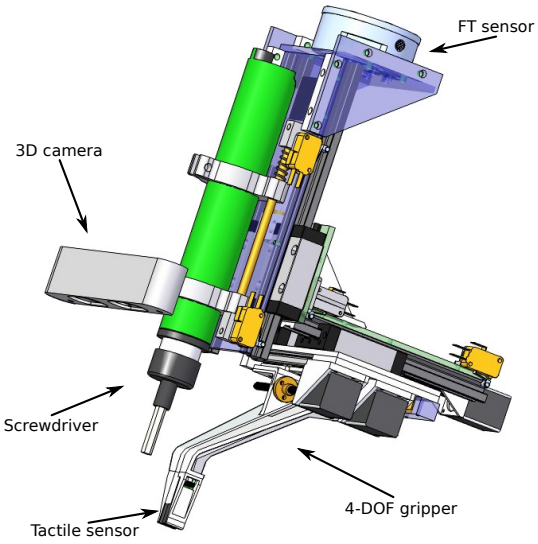


FIGURE 2.4: CAD of the final end effector developed for the WIRES project.

for all taxels. These displacements produce variations of the reflected light and, accordingly, of the photocurrents measured by the PTs. The taxel signals are acquired by a 16 channels ADC with Serial Peripheral Interface (SPI). The mechanical properties of the silicone cap determine the maximum load applicable to the sensor before cell saturation and, as a consequence, its sensitivity. The implemented sensor uses a shore hardness of 26 A, resulting in a maximum applicable force up to 30 N, with a sensitivity of about 0.3 N. An Arduino-based μcontroller board is then used to send the data to the control PC via USB connection.

The control system has been developed exploiting the ROS middleware to allow the communication between the different parts (sensors, robot, gripper, cameras, etc.) that compose the experimental setup.

Despite the gripper previously described is a preliminary solution, the end effector that will be adopted for the implementation of the whole cabling process is much more complex, see Figure 2.4. The whole end effector will integrate an FT sensor at the wrist interface, a 3D camera providing top view of the scene, an computer-controlled screwdriver (for the execution of phase 5)) and a 4-DOFs gripper (gripper opening and finger x-y-z position w.r.t. to the screwdriver tip) equipped with the aforementioned tactile sensor. In the final process implementation, the robot arm will be used to position the screwdriver tip on the terminal screw, and the FT sensor will be used to control the contact with the screw during the tightening. Therefore, the end effector will be held in an almost fixed position, just the screw motion during the tightening will be compensated. Consequently, the wire insertion (phase 4)) will be performed by using the gripper DOFs only. It results that the FT sensor cannot be used during the insertion and for the tightening check, because the magnitude of the force generated by the wire contact during phase 4) is much lower than the one generated by the contact between the screwdriver and the screw, making the former indistinguishable. In this work, the FT sensor has been used as ground through to train the tactile sensor integrated into the fingertip, as described in subsection 2.5.5.

Moreover, while the vision system can be easily applied during phase 1) and 3), it would be complex to adopt vision during phase 4) and 6) due to occlusion problems. In fact, several wires and components are usually present in the same

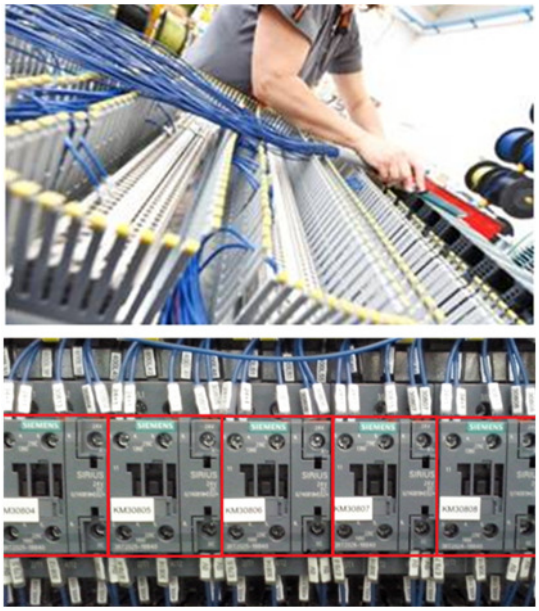


FIGURE 2.5: Different views of a switchgear during the assembly. The cables and components, often very similar and of the same color, make complex the detection of a specific terminal or wire from the scene.

scene, as shown in Figure 2.5. Moreover, even if a camera will be mounted in the end effector, the field of view of this camera will provide a top view of the cables and components, for the execution of step A) and phase 1), while during phase 3) the point of view of this camera will be ineffective. Additionally, during phase 4) the wire insertion point (i.e. the terminal) will likely be occluded by the component itself (since the terminals are usually located on two opposite sides of the component). The use of a stereo or 3D camera for phase 3) was discarded because, due to the relative position of the wire tip and the camera, the vision system must be explicitly designed for this phase to achieve the required precision and range of view. It follows that the same camera cannot be used for other operations such as step A) that requires a much larger vision field. Therefore, we opted for a 3D camera providing a top view of the scene with a vision field selected mainly for step A). In the final setup, a second 2D fixed camera will be placed close to the wire picking point (and not mounted on the end effector as in Figure 2.2). After phase 2), the robot will place the gripper in front this fixed camera to obtain a lateral view of the wire to execute phase 3). This solution do not increase the end-effector complexity, can be easily implemented since no space restrictions are present in the neighborhood of the wire picking region, is cheap and provide the better point of view to correct the most likely wire misalignment, i.e. in the gripper grasping plane.

2.5 System Pipeline

In this section, the self-labeling of the vision system training dataset by the robot will be introduced first, that represents the main novelty of the developed system. Secondly, the whole task execution pipeline as depicted in Figure 2.1 will be presented. Each of the four modules will be described in detail, highlighting the adopted machine learning algorithms along with the related training techniques.

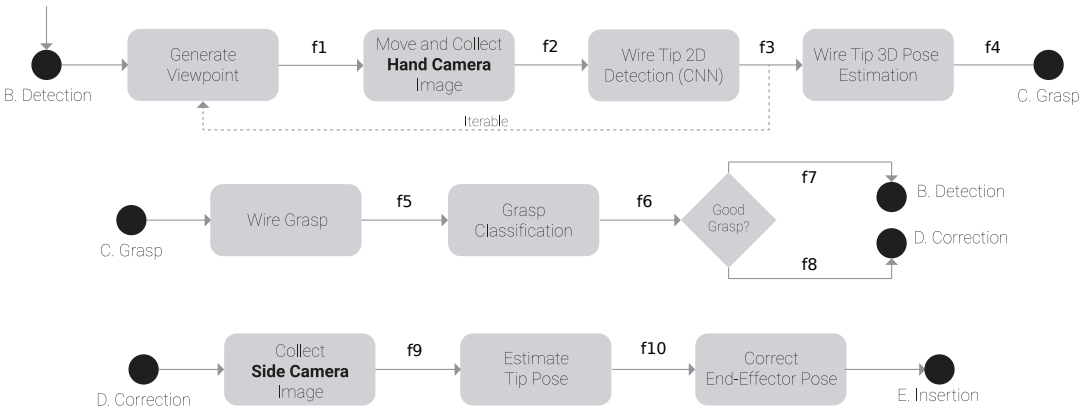


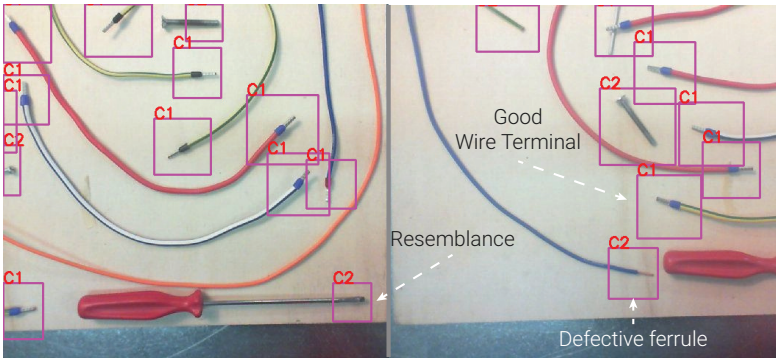
FIGURE 2.6: The overall Pipeline of our System resumed as an High Level Flowchart. The system is split into 3 sub-systems to facilitate its representation and labeled edges are depicted to understand better the dataflow. Each subsystem is described in detail in the related subsection: *B-Detection* in the subsection 2.5.2; *C-Grasp* in the subsection 2.5.3; *D-Correction* in the subsection 2.5.4 and *E-Insertion* in the subsection 2.5.5 (the latter is not represented as a detailed flowchart module because its simple nature, and furthermore it's the topic of a future research. The notation $m \in \mathbb{R}^{h \times w \times 3}$, introduced in this graph, represents a generic 3-channel image (e.g. an RGB image).

An overview of the pipeline is depicted in Figure 2.6. In this picture the system is presented with the flowchart metaphor to understand better the dataflow and the interconnections between subsystems. It should be noted that the *self-training* procedure, described in subsection 2.5.1, is not present because technically, as described later, it is functionally equal to the procedure described in subsection 2.5.2 and, moreover, it is an offline procedure not strictly related to the online pipeline.

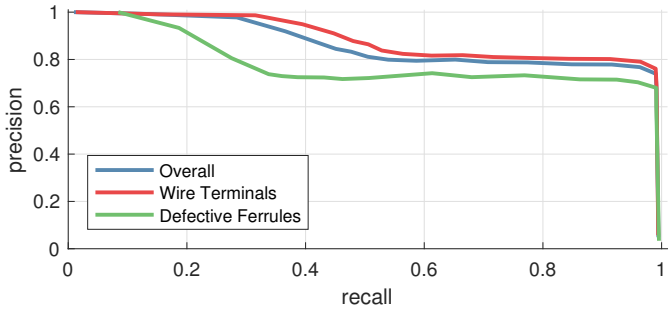
2.5.1 CNN Self-Training for Wire Terminal Detection

The major novelty of the vision system here developed is the self-learning procedure exploited by the robot to train the CNN for wire terminals recognition. A popular approach introduced in Mitash *et. al.* [50], that uses synthetic data (physics emulation), is not suitable for deformable objects like wires. Also the approach proposed by Georgakis *et. al.* [51] cannot scale enough to industrial environment due to the lack of public datasets outside the service robotics field. In our approach, a CNN [49] was trained to detect square-regions around the wire end in RGB images gathered by the hand camera (the camera mounted on the end effector). The adopted CNN also discerns good terminals from bad ones (e.g. bad-crimped ferrules). This CNN differs from a classical Region Based CNN (R-CNN) [52] because reframes the problem of detecting a square region and, at the same time, classifies the object within as a unique regression problem.

The CNN is not trained from scratch, instead a *Fine-Tuning* procedure is performed starting from the original network [49] pre-trained over the *ImageNet*[53] dataset. We trained the network for 100000 steps with a *batch size* of 24. In Figure 2.7a an example of self-labeled entries is presented. Our experimental dataset contains over 5000 RGB images, built in 4 hours (0.5 hours of human work + 3.5 hours of robot work). Considering an average time of 1 hour per 100 images, (estimated during our observations), the same labeling procedure done by hand would last at least 50 hours. With the term *labeled* image, we mean an image with square



(A) A couple of self-labeled random frames. This procedure generates more than *5000* labeled images in approximately *4 hours*. The same labeling task if had been entrusted to a distributed service like *Amazon Mechanical Turk*, minimizing user rewards, it would cost around *1000$*. Our labeling procedure trains the network to distinguish between the wire terminals (*Class1=C1*) and defective ferrules, or other resembling things (*Class2=C2*).



(B) Precision-Recall curves of the YOLO Object Detector ([49]) varying the output threshold. The mean average precision is 0.854.

FIGURE 2.7: Qualitative and quantitative results for the self-labeling procedure. The presented approach provides high performance with minimal human intervention.

regions drawn around target objects used to train a CNN to detect the same object/region in new unseen pictures. As for other machine learning approaches, also in this case the larger is the number of training samples, the better is the CNN recognition rate.

The solution here proposed exploits the inverse technique shown in Figure 2.8, that will be detailed in the next subsection. Given the position $\mathbf{p}_n$ of the wire terminals in homogeneous coordinates w.r.t. the robot reference frame, e.g. through measurement or by touching them with the end effector (this is the only human intervention), for every picture taken with the hand camera, it is possible to compute the projection y_n of $\mathbf{p}_n$ in the image coordinates using the pinhole camera equation

$$y_n = A \begin{bmatrix} R_{\text{cam}} & t_{\text{cam}} \end{bmatrix} \mathbf{p}_n, \quad A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

where A represents the intrinsic camera matrix (a camera dependent parameter). The matrix $\begin{bmatrix} R_{\text{cam}} & t_{\text{cam}} \end{bmatrix} \in \mathbb{R}^{3\times4}$ represents the poses of the hand camera in the world coordinate frame (i.e. the camera extrinsic parameters). This matrix can be computed given by the position of the robot end effector (provided by the robot) and the relative position between the end effector and the hand camera (known from the

end-effector design). In this way, given a scene with cables in known positions, it is possible to collect an arbitrary number of self-labeled training images just moving the hand camera around using the robot.

The outcome of this CNN is a set of rectangular frames, i.e. the "predictions", identified by both their 2D center coordinates x, y in the image plane, width w and height h, along with a label l identifying good ($l = 1$) or bad terminals ($l = 0$).

The Figure 2.7b shows Precision-Recall curves for the object detector used during experiments (YOLO [49]). The mean Average Precision (mAP) for the overall detection task is 0.854, with a 0.88 mAP for wire terminals and 0.77 mAP for defective ferrules. This precision gap is justified on the grounds that a wire terminal is higher in visual cues w.r.t. a simple defective ferrule. Considering that:

- the human intervention takes only 30 minutes;
- this labeling procedure needs to be repeated whenever the environmental conditions change or a new class of Wire Terminals is provided;
- in an industrial setting, the environmental conditions can be controlled and maintained fairly constant;
- just two classes of wire terminals are used in the actual industrial production;

the overall performance represents a suitable trade-off between functionality of the automatic system and required operator time. However, the proposed approach allows to easily generate training datasets in an extremely wide set of working conditions, far beyond the industrial scenario.

2.5.2 Wire Detection through CNN

It is worth mentioning that, in the real scenario, the wires to be connected into the switchgear will be produced and stored in a known region by an dedicated machine. Therefore, we can assume that the wires are arranged on a plane in such a way the gripper can grasp each wire without colliding with other wires or the environment. However, due to possible wire bends, the robot needs to locate the wire terminal and estimate its pose with sufficient precision to plan a correct grasp by using the hand camera.

This module exploits the output of the CNN described in the previous section to estimated the wire terminal 3D location in each image provided by the hand camera. This procedure is needed since the 3D hand camera resolution is not sufficient to achieve the desired wire grasp success rate. Thus, we need to reconstruct the 3D position of the target object, in this case the wire terminal, only exploiting multiple 2D information provided by the CNN. This is the well-known technique called Structure From Motion (SFM) [54], which exploits the triangulation algorithm to reconstruct the 3D coordinates of a 2D feature seen from multiple known vantage points. In the classical SFM approach, as well as in SLAM systems, the camera pose is computed simultaneously to the reconstruction phase. Unlike these approaches, we rely on the technique described in [55]. This technique exploits the robot high repeatability, along with a precise hand camera calibration, to compute in a closed form the 6-DOF pose of the hand camera. Theoretically, if we know the exact homogeneous coordinate y_n of our target object in the image I_n, two viewpoints only are needed to obtain a suitable triangulation result. However, in our case only a coarse region around our target terminal is available, as depicted in Figure 2.8. If the center of the square-region is chosen as 2D reference feature, a non-perfect overlapping with

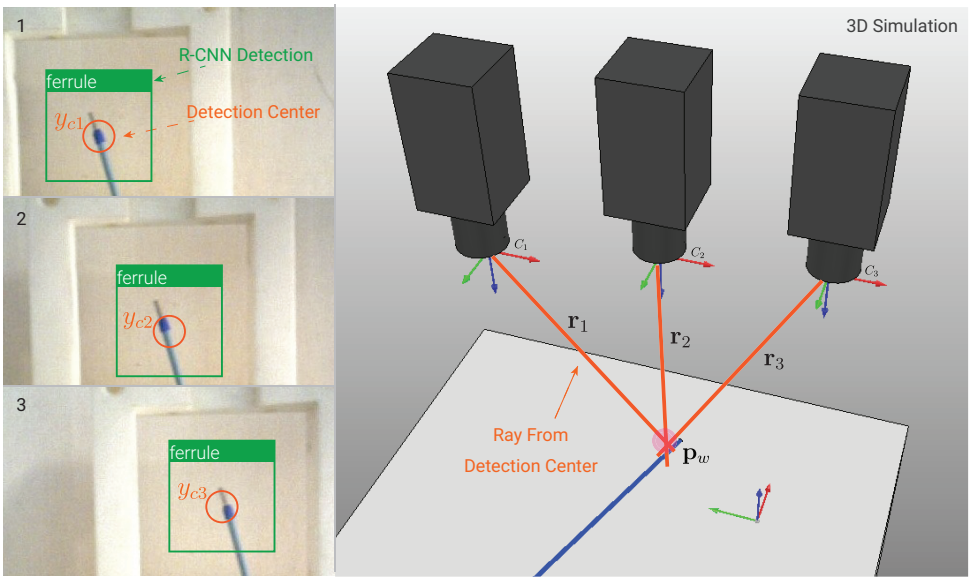


FIGURE 2.8: Depth estimation through multiple CNN detections. Three images were captured from the camera poses $C_{\{1,2,3\}}$ respectively, obtaining 3 different square regions through the "predictions". By projecting the rays $\mathbf{r}_{\{1,2,3\}}$ passing through each region's center $y_{c\{1,2,3\}}$, the wire terminal $\mathbf{p}_w$ is found. Conversely, knowing the wire terminal position $\mathbf{p}_w$ in the robot coordinate system and moving the hand camera using the robot, images from an arbitrary number of known camera poses C_n can be collected. Therefore, $\mathbf{p}_w$ can be projected into the images coordinates $\hat{y}_{cn}$, generating an arbitrary number of self-labeled images for the CNN training.

the tracked object center is obtained (this error strictly depends upon CNN architecture and is not treated in this work). Therefore, more vantage points are needed to achieve a more accurate 3D reconstruction. For each vantage point, a ray $\mathbf{r}_n$ can be computed, i.e. a unit vector in the camera reference frame, corresponding to the selected 2D feature:

$$\mathbf{r}_n = \left\| A^{-1} y_n \right\| \tag{2.2}$$

Together with the center of the camera frame $\mathbf{c}_n$, $\mathbf{r}_n$ generates a 3D line $l_n = (\mathbf{r}_n, \mathbf{c}_n)$. In the camera reference frame, $\mathbf{c}_n$ is zero, otherwise it represents the position of the hand camera in the world coordinate frame. Thus, given a set of lines l_n, the closest point $\mathbf{p}$, i.e. the point with minimum distance from all the lines (since a common intersection point could not exist with real measurements) can be computed by

$$\mathbf{p} = (\sum_i \mathbf{I} - \hat{\mathbf{r}}_i \hat{\mathbf{r}}_i^T)^{-1} (\sum_i (\mathbf{I} - \hat{\mathbf{r}}_i \hat{\mathbf{r}}_i^T)\, \mathbf{c}_i)$$

where $\hat{\mathbf{r}}_i$ is any perpendicular unit vector to $\mathbf{r}_i$ [56].

In Figure 2.9 the results of the proposed algorithm scanning three different wires (in color and dimension) with different camera movements (parallel and orthogonal to the image plane) are presented. The achieved precision is approximately 1 cm collecting more than 20 images from different viewpoints. At the same time our results show that the best performance is achieved by moving the camera parallel to image plane. These experiments are designed as a proof of the approach correctness and not to evaluate best hyper-parameters to reduce the error (i.e. the optimal distance between camera and target object). To generalize the error in the depth estimation,

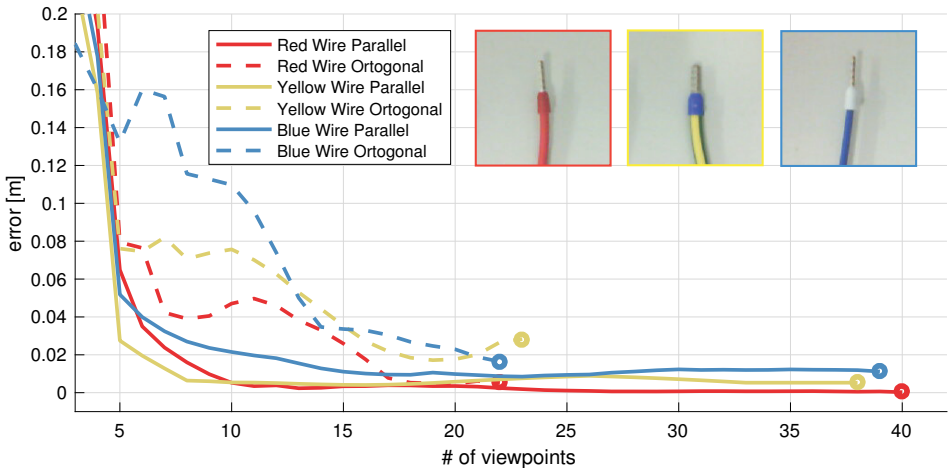


FIGURE 2.9: Wire terminal depth estimation error w.r.t. the number of images collected from different viewpoints moving the hand camera by using the robot. Three wire terminals, *Red*, *Yellow* and *Blue*, and two motion directions, *Parallel* or *Orthogonal* to the plane on which the cable lies, were tested. The *Parallel* movement provides better results since it minimizes the likelihood between the rays, achieving an error lower than 1 cm after 10÷15 images.

the Stereo Vision error formulated by Gallip *et. al* [57] can be used

$$\epsilon_z = \frac{z^2}{b \cdot f} \epsilon_d$$

where ϵ_z is the depth estimation error, z is the distance, f is the focal length of the camera, b is the baseline (e.g. the distance between the two camera, in a classical stereo vision setup, or the two vantage points in our system) and e_d is the disparity error. Since the 6-DOF camera pose can be controlled by moving the manipulator, it is possible to select the couple (z, b) to reduce the error ϵ_z according to the circumstances. Now, we can define a generic function:

$$\tau(y_0, ..., y_n, C_0, ..., C_n) = \mathbf{p} \tag{2.3}$$

to compute the 3D position $\mathbf{p}$, corresponding to the 2D feature y using multiple images.

Unfortunately, just one 2D feature is not enough to infer a 3D reference frame associated with the wire terminal. Therefore, at least 2 features are needed to estimate a 3D vector corresponding to the final part of the wire by means of the Equation 2.3. In Figure 2.10 the algorithm used to infer a 2D reference frame H_w of the wire terminal is shown starting from a square region of the image, in a nutshell: starting from the image 1) an adaptive threshold is applied to the target region obtaining in 2) a binary image enhancing wire's pixels and removing background; 3) the region is rotated w.r.t. a custom reference frame $H_\perp$ placed on the mid point of one side of the frame, chosen such that it is the nearest point to the intersection of the wire with the square region; 4) an *orthogonal regression* is applied to the binary image to estimate the best fitting line $L = \{(x, y) \mid y = mx + q\}$ onto wire points; use the rightmost wire's pixel projection onto line as the center of H_w and the angle of L as its orientation, i.e.

$$H_w = (y_w, \theta_w) = (y_w, \tan^{-1}(m)) \tag{2.4}$$

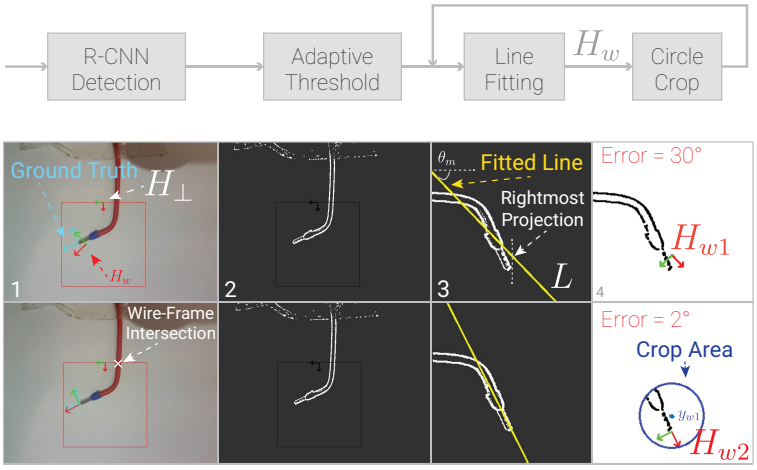


FIGURE 2.10: 2D wire terminal reference frame estimation. Each row represents a pipeline iteration: *1)* CNN detection over the wire image; *2)* binary output obtained by the adaptive threshold; *3)* the detected region is rotated w.r.t. $H_\perp$ and the fitting line is superimposed; *4)* the wire reference frame H_{w1} is laid out. In the second row, another iteration is performed carving out a circular region around y_{w1} and repeating the line fitting process.

In Figure 2.10 a case in which the cable is strongly bent is shown: this situation is useful to show the effectiveness of the proposed algorithm but it is unlikely in a real scenario. This sequence can be considered as the worst case, in which the ferrule is almost orthogonal to the portion of the cable intersecting the square region. The proposed algorithm produces a quite inaccurate wire end pose after the first iteration in this case. However, it is possible to see how a second iteration applied to a cropped region around the previous estimated center y_w ensures that line fitting is relative only to the terminal part and not to the whole wire within the region. In subsection 2.5.4 the way how this algorithm is applied to the images provided by the side camera to estimate the wire end reference frame for pose correction will be further explained. Here, instead, we can just take advantage of the center homogeneous coordinate y_w of H_w and use Equation 2.3 to estimate its 3D position in the robot reference frame. Thus, from the pairs y_w, y_c, where y_c is the homogeneous coordinate of the center of the aforementioned CNN detection, see Figure 2.8, collected from multiple viewpoints, we can compute their corresponding 3D points

$$\mathbf{p}_w = \tau(y_{w\{1,\dots,n\}}, C_{\{1,\dots,n\}}),\ \mathbf{p}_c = \tau(y_{c\{1,\dots,n\}}, C_{\{1,\dots,n\}})$$

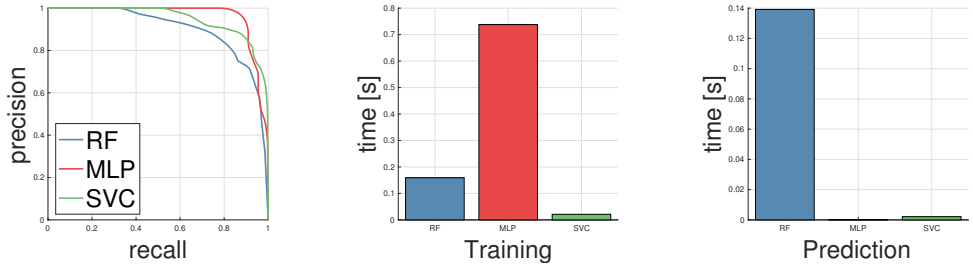
Thus, given the pair $(\mathbf{p}_w, \mathbf{p}_c)$,

$$\mathbf{v}_{w_x} = \frac{(\mathbf{p}_w - \mathbf{p}_c)}{\|\mathbf{p}_w - \mathbf{p}_c)\|}$$

represents the unit vector oriented along the wire terminal symmetry axis, while

$$\mathbf{v}_{w_z} = \mathbf{v}_{w_x} \times \pm\mathbf{u}_x,\ \mathbf{u}_x = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$$

(A) The binary classifier distinguishes between *Good* (1) and *Bad* (0) grasps using tactile sensor measurements.



(B) Classification metrics over RF, MLP and SVC. MLP shows better results and the lowest prediction time.

FIGURE 2.11: The wire grasp classifier. This classifier was self-trained over the outcomes of the system module described in subsection 2.5.4 choosing – by design – the thresholds that would affect the rest of the task (e.g. an escaping terminal shorter than 1 cm is considered a *bad* grasp).

indicates the forward direction w.r.t. the robot reference frame. Therefore, the pose ${}^0\mathbf{T}_w$ of the wire end can be defined as

$$ {}^0\mathbf{T}_w = \begin{bmatrix} \mathbf{v}_{w_x} & \mathbf{v}_{w_x} \times \mathbf{v}_{w_z} & \mathbf{v}_{w_z} & \mathbf{p}_c \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_w & \mathbf{p}_c \\ 0 & 1 \end{bmatrix} \tag{2.5} $$

Note that the orientation $\mathbf{R}_w$ is chosen by construction considering that the wire terminal is symmetric along its $\mathbf{v}_{w_x}$ axis. Thus, the $\mathbf{v}_{w_z}$ component, can be arbitrarily chosen to point toward the ceiling. To this end, the sign of $\mathbf{u}_x$ is chosen case-by-case to avoid that $\mathbf{v}_{w_z}$ points to the floor instead. The 6-DOF pose of the wire ${}^0\mathbf{T}_w$ is then used by the robot in order to perform the grasp.

2.5.3 Wire Grasp

In order to facilitate the insertion, the wire should be grasped at the center of the gripper fingers. It is important here to recall that, both in the real scenario and during the experiments here reported, the wire lays on a plane with its terminal section in the free space. This allows the gripper to grasp the wire without colliding with other wires or the environment.

Even though the module described in subsection 2.5.4 can be used to detect if the wire as been effectively grasped or not, there will be no way to recover from a grasp failure at this stage, mainly because it will be very challenging to perform a re-grasp once the wire has been removed from its docking position. For this reason, the outcome of the wire pose detection module described in subsection 2.5.2 is exploited to train a classifier able to detect if the cable is in a suitable pose w.r.t. the fingers from tactile sensor data only. Therefore, the use of the tactile sensor allows to evaluate immediately the grasp correctness, removing in this way the possibility of reaching unrecoverable situations. Three different machine learning algorithms for classification are tested and compared: a Multi-layer Perceptron Neural Network (MLP) with 3 hidden layers composed of 16, 8 and 2 neurons; a Random Forest (RF) with

200 trees; and a Support Vector Classifier (SVC) with a radial basis function kernel. Figure 2.11 shows some example of *good* and *bad* grasps together with some classification benchmark among different classifiers trained during our experiments. The generic classifier is fed with $x_t \in \mathbb{R}^{16}$, representing the 4×4 matrix of the tactile measurement, coupled with $\hat{h} \in \{1, 0\}$, that is a boolean information representing if the wire grasp configuration is within an admissible range or not respectively. Taking into account Equation 2.4, the parameter $\hat{h}$ is defined as

$$\hat{h} = \begin{cases} 1 & l_{\min} \geq \|{}^{\perp}y_w\| \geq l_{\max} \wedge \theta({}^{\perp}H_w) \leq \theta_{\max} \\ 0 & \text{otherwise} \end{cases}$$

where $\theta({}^{\perp}H_w)$ means the orientation of the 2D reference frame ${}^{\perp}H_w$ (i.e. is H_w expressed in $H_{\perp}$) and $\theta_{\max}, l_{\min}, l_{\max}$ are the parameters defining the terminal orientation/position admissible range with respect to the fingers. The results shown in Figure 2.11(b) were obtained with a dataset of over 200 grasp samples. The best performance is provided by the MLP algorithm. However, since the purpose of this classifier is to ensure that *bad* grasps are detected, the objective is to maximize *precision* and not *recall*. Therefore, any of the evaluated classifiers can be used for this problem because all reach precision equal to 1 in the *precision-recall* curve. In case of bad grasp is detected, the gripper is retracted without removing the wire from its docking, and the procedure restarts from the wire pose detection. Increasing the number of viewpoints can be used in this case to possibly reduce the wire pose estimation error.

2.5.4 Wire Pose Correction

In this stage of the pipeline, the system aims to estimate the pose of the wire w.r.t. the gripper by means of the side camera framing laterally the fingers, as depicted in Figure 2.12. This problem is a simplification of the one seen in the subsection 2.5.2. Indeed, here the pose of the side camera ${}^{ee}\mathbf{T}_{cam}$ w.r.t to the gripper is known by construction. Therefore, the pose of the side camera in world coordinates can be easily computed as ${}^{0}\mathbf{T}_{cam} = {}^{0}\mathbf{T}_{ee}\,{}^{ee}\mathbf{T}_{cam}$, where ${}^{0}\mathbf{T}_{ee}$ is the actual forward kinematics solution. In case a fixed camera is used to that purpose, the camera position ${}^{0}\mathbf{T}_{cam}$ is known and the camera position w.r.t. the end effector ${}^{ee}\mathbf{T}_{cam}$ can be compute inverting the previous formula. A mandatory step of this module is to calibrate correctly the pose of the side camera ${}^{0}\mathbf{T}_{cam}$ in the robot coordinate system. To perform a correct extrinsic calibration, the method seen in [55] is exploited by means of an *Augmented Reality* marker [58] printed in a known pose w.r.t. the fingers of the gripper. The marker attached to the back side of finger is visible in Figure 2.12. This approach allows to calibrate the side camera pose on-line, enabling to use a moving camera instead of a fixed one (e.g. camera mounted on another robot).

By exploiting the knowledge of the camera and the gripper position, the distance of the grasp plane (i.e. the plane on which the wire lies after grasp) can be computed in analytical form as $d_w = {}^{cam}p_{z_{ee}}$, where ${}^{cam}p_{z_{ee}}$ is the z component of the translation part of ${}^{cam}\mathbf{T}_{ee}$. Then, given the *depth* d_w of the image pixels, the conversion from homogeneous to 3D coordinates (in the camera reference system) can be computed as

$$\pi(y) = \pi(\begin{bmatrix} u & v & 1 \end{bmatrix}^T) = \mathbf{p} = d_w \begin{bmatrix} \frac{(v-c_x)}{f_x} & \frac{(u-c_y)}{f_y} & 1 \end{bmatrix}^T \tag{2.6}$$

where $y = \begin{bmatrix} u & v & 1 \end{bmatrix}^T$ are the homogeneous coordinates of a generic pixel in the 2D image, $\mathbf{p}$ is the corresponding 3D point, c_x, c_y, f_x and f_y are the parameters of

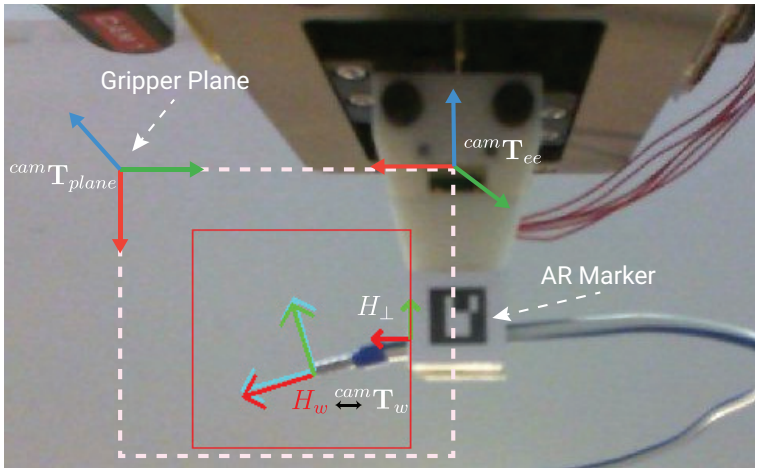


FIGURE 2.12: Estimation of the wire terminal pose after grasp. The flattest part of the finger contains the *Augmented Reality Marker* used to calibrate the camera 6-DOF pose related to the robot reference frame.

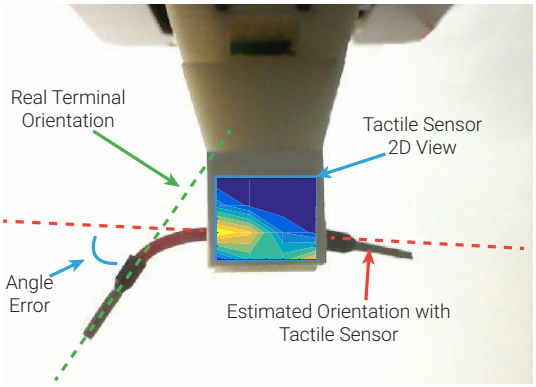
the camera matrix A. Thus, the systems exploits the same algorithm seen in subsection 2.5.2 (see Figure 2.10). The only difference is that, in this case, the reference frame $H_{\perp}$ is chosen to be as close as possible to the fingers' center. The whole procedure is shown in Figure 2.12. Hence, given the position of the wire terminal w.r.t. the end effector $^{ee}\mathbf{T}_w$, the wire terminal position in the world coordinates is $^{0}\mathbf{T}_w = {}^{0}\mathbf{T}_{ee}\,{}^{ee}\mathbf{T}_w$.

In Figure 2.13 the error rate on the 2D wire pose estimation after the grasp varying the tilt of the cable is reported. In Figure 2.13(b) we can see how, choosing a desired *crop size e.g.* between 1 and 2, the estimation error is under 5 pixel for the position and 5 deg for the orientation (the dotted black line) considering a wire tilt angle in the range ± 45 deg (i.e. for the first three curves in the legends).

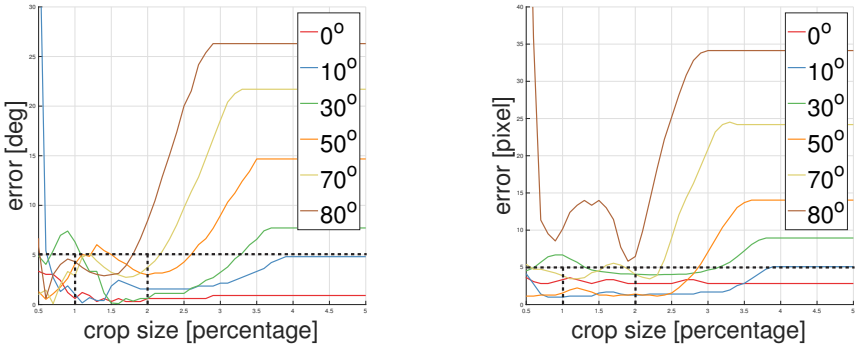
Obviously, the error metric derived from the pixel error is proportional to the distance between the camera and the target object; for a distance of 0.1 m we have a conversion factor of $k = 0.0002$, then 5 pixel = 0.0001 m error. As presented in Figure 2.13 we can predict, with the side camera at a distance of about 0.15 m, an average error on the estimated terminal position under 1 cm, and below 10 deg for the orientation. This tolerance is permissible because we can correct the end-effector pose to align the wire terminal and perform a correct insertion. However the insertion task may fail (despite the further pose correction module) in the following possible situations: 1) The wire is too close to the finger bottom edges (possible loss during the transport); 2) The orientation is impossible to correct by a twist of the end-effector (gripper kinematic limitations).

2.5.5 Wire Insertion

After the wire pose correction accomplished during the previous stage, the insertion task can be executed by planning a trajectory of the wire terminal frame $^{0}\mathbf{T}_w$ toward the component hole. This task can be seen as a *peg-in-hole* problem. In this phase, machine learning is exploited to infer from the tactile sensor data the same information coming from the FT sensor. This is needed to detect impact between cable terminal and the component, and eventually to correct the wire trajectory during the insertion into the terminal. Normal and tangential forces components can be



(A) 2D view of the tactile sensor is superimposed over a real RGB image of a grasped wire. The red dashed line represent the best fit provided by the tactile sensor.

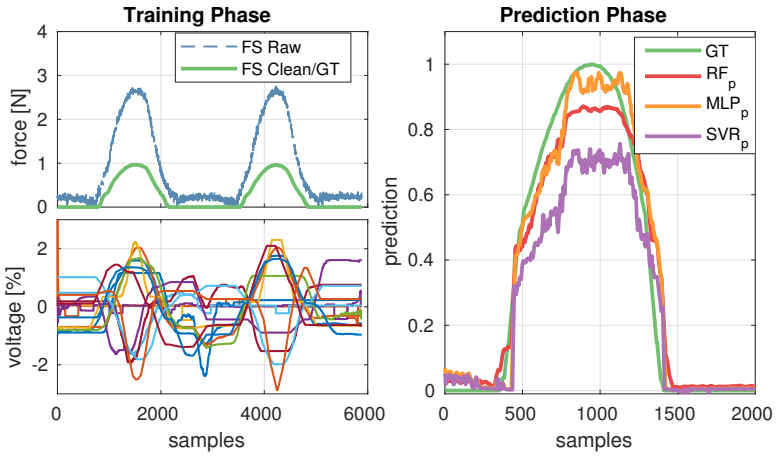


(B) Estimation error varying the angle of the wire terminal w.r.t. the prediction of the tactile sensor. The x-axis reports the crop radius normalized over the ferrule size (in pixel).

FIGURE 2.13: Detection of the pose of a grasped wire by the vision system.

distinguished by the tactile sensor described in [48] since they are related, respectively, to symmetric and asymmetric variations of the measured pressure map. As a consequence, during a collision, a strong correlation exists between the direction of movement of the grasped object and the signal pattern x_t provided by the 4×4 taxels matrix. For this reason, x_t is exploited to train a regressor able to provide a scalar continuous variable representing the magnitude of the collision force in the tactile sensor plane.

Aiming at comparing alternative solutions, several regressors are trained by the robot itself, collecting data during many collisions between a flexible barrier and a grasped terminal in known pose (as provided by the wire pose correction module). These data are used to predict a real value associated with the impact force and quantify the latter in a continuous manner. In Figure 2.14 the data used during the training procedure involving tactile and force sensor data are shown. From this figure, it is clear that the MLP produces a suitable prediction of the contact force. Figure 2.14 shows a Mean Square Error (MSE) and Dynamic Time Warping (DTW) analysis [59] performed over the *regressor*. The regressor is trained over 15000 samples collected during controlled impacts with different cable diameters in different angle w.r.t. the fingers. During training, the wrist-mounted FT sensor is used as ground truth. From these results, it is possible to conclude that the insertion task can be monitored exploiting the tactile sensor only, without using a vision system or a more expensive FT sensor.



	RF	**MLP**	SVR
MSE	0.0085	**0.0066**	0.0124
DTW	0.0264	**0.0182**	0.0347

FIGURE 2.14: *Training* and the *Prediction* phases of the collision detection regressor. On the left plot, a couple of impacts measured by both the force sensor (top) and the tactile sensor (bottom); the green curve is the normalized output of the force sensor used as training reference. On the right plot, the execution of the predictors RF (Random Forest), MLP (Multi-Layer Perceptron) and SVR (Support Vector Regression) speculating about terminal collisions. On the bottom table, the performance of the predictors in terms of *MSE* (Mean Square Error) and *DTW* (Dynamic Time Warping [59]).

A *Regressor* is adopted during the wire insertion to quantify the collisions instead of a simple *Classifier* that can simply detect it. By exploiting this approach, it is possible to distinguish between actual collisions, as shown in Figure 2.17a, from rubbing of the wire end inside the component terminal hole, as reported in Figure 2.17b. While in the former case the insertion task is stopped and completely replanned, in the latter one the task is very likely to be successfully accomplished. In chapter 4, we extend this step by exploiting the regressor output as a feedback signal to guide the insertion task. The aim is to increase the insertion success rate even in case of lateral contact between the wire end and the component terminal hole.

2.6 Evaluation of the Insertion Task Sequence

In Figure 2.15 a sequence showing the wire pose correction and insertion into the component simulacrum hole is shown. Starting from a common configuration (1), in the top sequence a wrong alignment is provided to the system at step (2) to shown the effect of collisions. It results that the wire terminal is not aligned with the reference line and, as a consequence, with the hole at stage (3), causing a collision at stage (4). In Figure 2.17a the artificial wrong wire terminal orientation, the robot x-axis position and the filtered output of the MLP regressor over tactile sensor during the wire pose correction and insertion task are reported for the wrong sequence in Figure 2.15. Looking in particular at the MLP output, the effect of the collision measured by the tactile sensor can be clearly seen in the final part of the insertion

TABLE 2.1: Wire insertion results for a cable with external diameter of 2.0 mm on the left and 3.5 mm on the right. Parameters m and d refer to initial condition of the wire w.r.t. the gripper, while c refers to the result of insertion where $c = 1$ is a positive outcome.

#	m [deg]	d [mm]	c	#	m [deg]	d [mm]	c
1	-5.7	48.0	1	1	13.5	27.0	1
2	-1.0	32.5	1	2	-20.3	30.0	1
3	4.0	38.9	1	3	4.6	29.4	1
4	-2.9	27.6	1	4	4.0	30.0	1
5	27.5	46.5	1	5	-12.4	29.0	1
6	-32.6	40.6	0	6	23.3	43.0	0
7	-15.6	38.5	1	7	-19.8	41.0	1
8	12.4	39.4	1	8	0.6	40.0	1
9	14.0	29.5	1	9	24.2	48.0	0
10	10.8	24.4	0	10	21.8	52.0	1
11	46.4	56.0	0	11	52.9	56.0	0
12	42.9	60.9	1	12	7.4	35.6	1
13	-28.4	45.3	0	13	47.5	52.0	1
14	41.3	65.9	0	14	58.6	95.0	0
15	33.0	39.0	1	15	44.7	69.2	0

phase. The bottom sequence, instead, shows how the vision system allows to correctly align the wire terminal with the reference line at step (2) and, consequently, with the insertion hole at step (3). It results that the wire is correctly inserted into the hole at step (4). Figure 2.17b shows the wire terminal orientation, position and the MLP output over the tactile sensor data during the wire pose correction and insertion task reported in the good sequence of Figure 2.15. After the initial estimation of the wire terminal pose, the end-effector orientation is corrected and the insertion task is executed: during the insertion, the MLP output allows to detect a contact between the wire terminal and the internal part of the hole causing friction. The output of the MLP regressor is not considered during the other phases to avoid false positive.

The performance of the overall pipeline were evaluated by executing the whole task for about 30 times with a wide range of working conditions. In Table 2.1 two subsets of 15 runs executed with two wires which external diameter is 2 mm and 3.5 mm, respectively, are reported. In these tables, m is the estimated wire terminal orientation, d denotes the distance of the wire tip from the finger center and c is 1 in case of successful insertion or 0 in case of failure. With reference to Figure 2.17

$$c = \begin{cases} 0, & \text{Position} > 0.65 \wedge \text{MLP output} > 0.5 \\ 1, & \text{Position} > 0.65 \wedge \text{MLP output} < 0.5 \end{cases}$$

where the position threshold means that the wire terminal is inserted. The data reported in Table 2.1 include also experiments performed in extreme conditions to test the system robustness. It results a overall success rate of about 66%. Looking at these results in the $\{m, d\}$-plane, as reported in Figure 2.16, it is possible to define an admissible working region of $m = \pm 20$ deg and $d \leq 50$ mm containing almost only successful wire insertions, 15 over 16, resulting in a success rate of about 95%. This working region can be easily addressed in the partially structured application

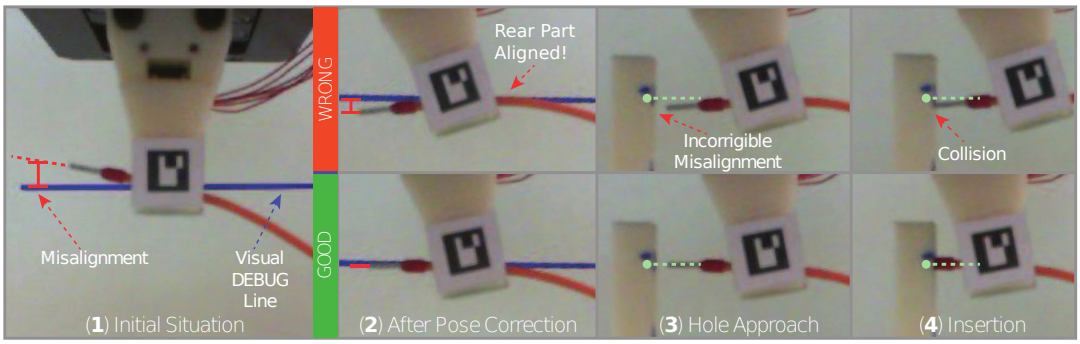


FIGURE 2.15: Evaluation of the whole insertion task. Collision detection exploiting tactile feedback in case of wrong alignment (top sequence) and wire terminal pose correction and insertion using the vision feedback (bottom sequence).

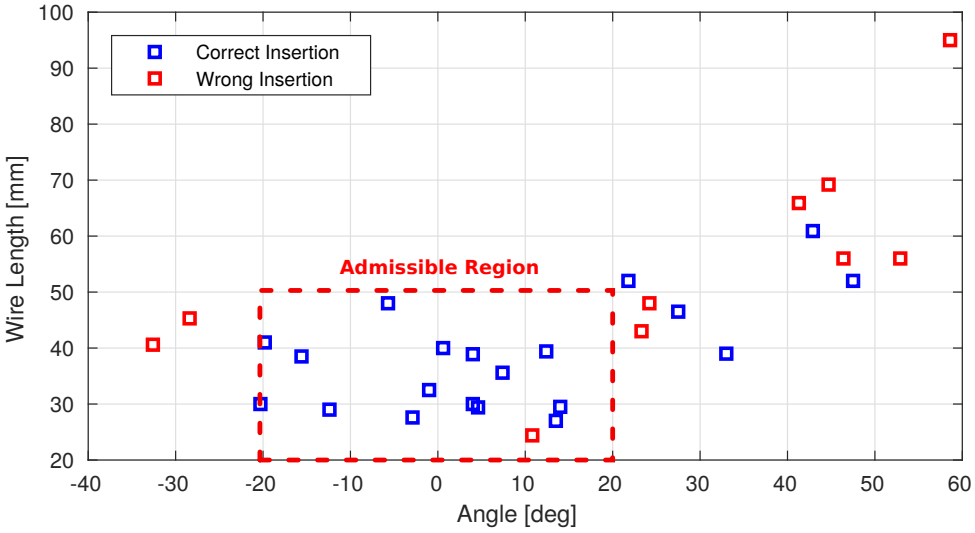


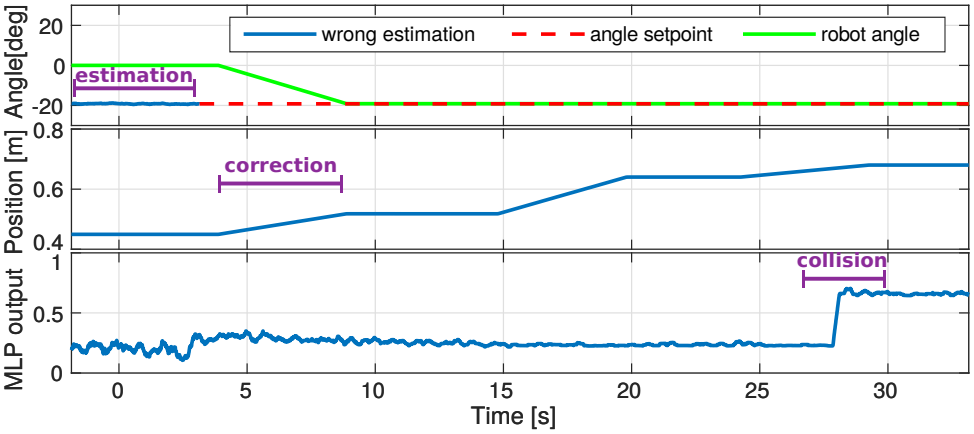
FIGURE 2.16: Evaluation of the admissible operating range of the developed wire manipulation and insertion system.

scenario.

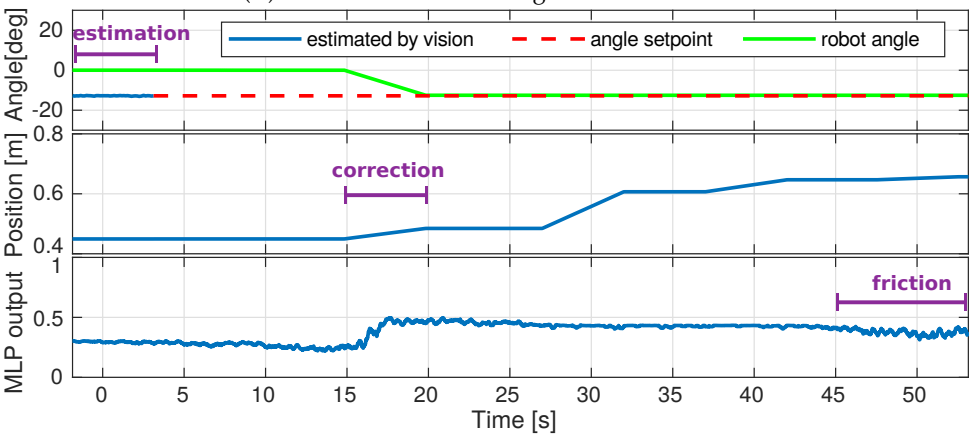
A qualitative evaluation for each building block of the insertion pipeline is shown in the supplementary material.

2.7 Conclusions

In this chapter, we described a system that performs wire detection, grasping and insertion into a component hole using suitable combinations of vision and tactile feedback. The synergy between cameras and tactile sensors allows to deal with the typical issues in the switchgear wiring scenario. Several machine learning algorithms are exploited for the system development, both for the vision and for the tactile module. Moreover, suitable techniques are developed for the automatic generation of the training datasets, allowing to significantly speed up the implementation of the target application.



(A) Collision detection using the tactile sensor.



(B) Wire pose correction and insertion using the vision system.

FIGURE 2.17: Execution of the wire pose correction and insertion tasks.

Chapter 3

Self-Supervised Learning for 3DoF Pose Estimation

In this chapter we further look into the technique presented in subsection 2.5.1 for generating the dataset that we used to train a wire-terminal detector. We show that the proposed method can be effectively applied, not only to wire terminals lying on an homogeneous white background, but also to other objects with different amount of visual features on both simple and complex backgrounds.

Moreover, we propose a strategy that leverage on classical CNN-based Object Detectors for estimating the angle of a target object as a classification problem. The resulting algorithm, that we named *LOOP* [60], can readily replace the hand-crafted solution used in subsection 2.5.2 and subsection 2.5.4 to infer the 2D reference frame of the wire terminal, bringing an improvement to the entire system reliability.

3.1 Introduction

For relevant robotic applications, such as, for instance, a fully automatic pick-and-place, the Pose Estimation from visual data is an essential stage of the overall pipeline. Unfortunately, the availability of training data hinders deployment of Convolutional Neural Networks (CNNs). In fact, we observe that while 2D Object Detection with category-level classification has achieved great effectiveness thanks to CNNs [49],[61],[62], the more complex perception task of the 3D Pose estimation has not experienced the same strengthening, notwithstanding remarkable results [63],[64],[65] have endorsed CNNs also for this task. We argue that between the key reasons for this state-of-affairs is the lack of training data: while for 2D Object Detection huge datasets like Pascal VOC [66] or COCO [67] define a reliable testbed for the community, the same can not be said for the 3D counterpart (with the exception of some small datasets *e.g.* like [68],[69]). Thus, the claim of our work is that for a real industrial application it is not sufficient to develop advanced data-driven models, like a convolutional neural networks, but – simultaneously – the data sourcing problem should be addressed. Thus, we propose to tackle the object detection and 3-DoF pose estimation task by an integrated framework based on CNNs wherein the required labeled training data are sourced autonomously, *i.e.* with negligible human intervention.

The basic idea of our approach is to take advantage of the known difficulty of CNNs to learn rotational invariant image features. As shown in [70],[71] these networks redundantly learn multiple representation of sought objects when they exhibit multiple rotation in the training images. As opposed to approaches like

[72],[70], which try to learn rotation-invariant representation, we leverage on classical CNN-based Object Detectors to formulate the angle estimation task as a classification problem by leading the network to interpret each single object orientation as a stand-alone class (for this reason we name the algorithm **LOOP**: **L**everaging on a generic **O**bject detector for **O**rientation **P**rediction). As previously mentioned, we endow our approach with an automated dataset generation technique that allows to label an entire video sequence easily, provided that the sequence features the same type of image acquisitions conditions as those in which the detector is exptected to operate at test time (*e.g.* a quite-planar scene in front of the camera). This approach enables collections of massive amounts of training data which, in turn, allow the creation of an almost perfect object detector (*i.e.* ~ 0.99 *mAP* according to our experiments). An open source implementation of the proposed method is available online [1].

3.2 Related Works

Nowadays, object detection and 3-DoF Pose Estimation in industrial settings is mainly addressed by classical computer vision approaches based on hand-crafted 2D features, which can effectively represent the orientation of salient local image structures. SIFT [73] is one of the most popular 2D feature detector and descriptor by which it is possible to implement a full Object Detection pipeline for textured objects. By matching multiple 2D local features it is possible to estimate the homography (or even a rigid transform) between a model image and the target scene. SIFT can be replaced by other popular alternatives, like SURF, KAZE, ORB, BRISK etc.. [74] presents a comprehensive evaluation of the main algorithms for 2D feature matching. As stated before, the aforementioned methods are suitable – mostly – for *textured* objects detection, but the same matching pipeline can be deployed replacing them with detectors/descriptor based on geometrical primitives (*e.g.* oriented segments) amenable to texture-less objects. One of the leading *texture-less* object detectors is BOLD [75], which was then followed by BORDER [76] (and its extension, referrred to as BIND [77]). A popular alternative to features, instead, is Rotation-Invariant Template Matching like OST [78], OCM [79] or Line2D [80]. However, a tamplate-based approach may not be the most efficient solution for real-time applications.

The above-mentioned considerations have lead us to compare LOOP mainly with SIFT [73] and BOLD [75], undoubtedly two state-of-the-art approaches in *textured* and *texture-less* object detection, respectively. Our claim is to propose a real-time deep learning alternative able to cope with both textured as well as untextured models and, seamlessly, over plain and cluttered backgrounds. As stated in section 3.1, as we can exploit any generic CNN Object Detector, we investigated here about well established approaches like YOLO [49] and SSD[61], which resolve the 2D object detection and recognition task with a single image analysis pass, as well as Faster R-CNN[62], which instead conceptually splits the detection and recognition parts. We found that YOLOv3 [81], the newest declension of the classical YOLO algorithm, represents a satisfactory test case for our experiments, though is worth pointing out that LOOP is detector-agnostic.

Another research line to attain robotic grasping systems concerns direct estimation of grasping points from images by means of CNNs. One of the most used approach, conceptually similar to our method, is the 2D Rectangular Representation

[1] https://github.com/m4nh/loop

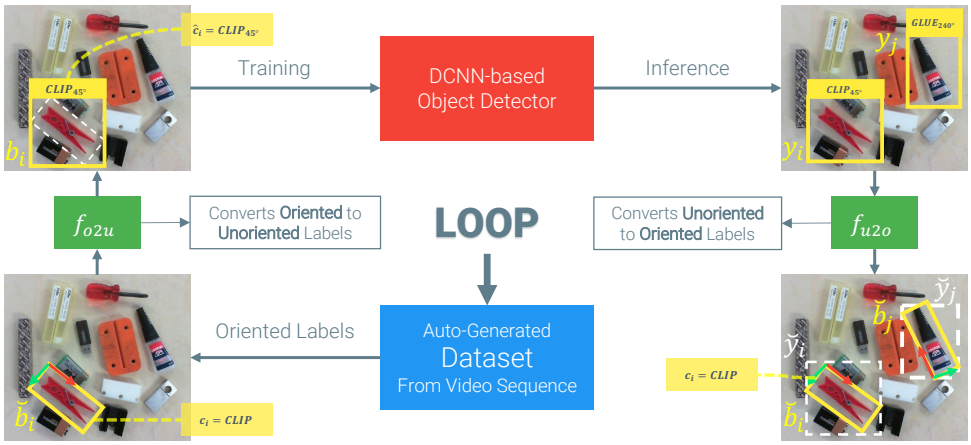


FIGURE 3.1: The overall pipeline of LOOP. The starting point is the creation of a dataset of oriented bounding boxes (oriented labels) with minimal human intervention. The *oriented* labels are converted in *unoriented* labels, by means of the f_{o2u} function , suitable to train a classical object detector, which encodes objects orientation in the classification process. Hence the detector infers *unoriented* predictions (y_i), with the same orientation encodings. The *unoriented* predictions are then transformed in *oriented* predictions ($\breve{y}_i$) by means of the f_{u2o} function.

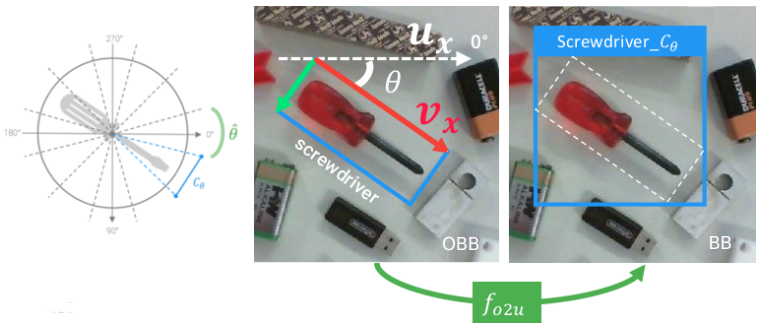


FIGURE 3.2: Graphic representation of the conversion between *Oriented-to-Unoriented* bounding boxes. The real angle θ is converted in the index C_θ of the corresponding quantized bin among the $k = \lceil \frac{2\pi}{\hat{\theta}} \rceil$ possible bins.

of grasp, as described in [82]. The authors demonstrated that a 2D representation of grasp is enough to perform a 3D manipulation with a robotic arm. Recent works like [83] or [84] estimate the position and the orientation of these 2D Rectangular Representation of grasp by means of a CNN, as either a regression or classificatiton problem, respectively. However, we believe that the full 2D Oriented Bounding Box of the sought objects yielded by our approach is a better representation for grasp in planar setting, because it allows to perform both obstacle avoidance as well as model-based grasp points computation.

3.3 LOOP: Leveraging on a generic Object detector for Orientation Prediction

As illustrated in Figure 3.1, given an RGB image, the LOOP framework can produce a set of predictions $\breve{y}_i = \{\breve{b}_i, \theta_i, c_i\}$, where $\breve{b}_i = \{x, y, w, h\} \in \mathbb{R}^4$ represents the

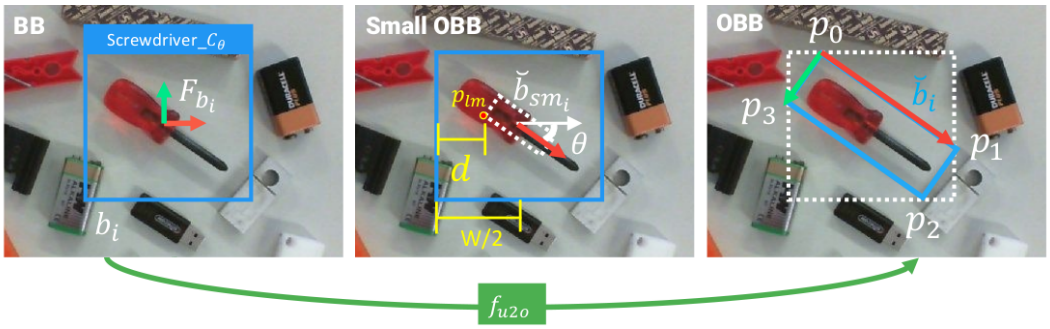


FIGURE 3.3: A graphical representation of the conversion between *Unoriented-to-Oriented* bounding boxes. It is important to know the *ratio* of the sought object in order to produce a commesurate oriented bounding box.

coordinates of the Oriented Bounding Box (OBB in short) clockwise-rotated by an angle of θ_i and $c_i \in \mathbb{Z}^+$ is the object class. As already mentioned, we leverage on a classical object detector, which outputs a set of simpler predictions $y_i = \{b_i, \hat{c}_i\}$ where $b_i = \{x, y, W, H\} \in \mathbb{R}^4$ represents the coordinates of the *unoriented* Bounding Box (BB in short) and $\hat{c}_i \in \mathbb{Z}^+$ encodes, with our formulation, both the object class as well as orientation information. In subsection 3.3.1 we will explain how to transform an oriented prediction $\breve{y}_i$ into an un-oriented one, y_i, while in subsection 3.3.2 we will describe the inverse procedure. Finally, in subsection 3.3.3 we will explain how to generate Oriented Bounding Box labels for an entire video sequence by labeling just the first frame.

3.3.1 f_{o2u}: The Oriented-to-Unoriented Function

As stated before, our approach is an extension of a classical 2D Object Detector to make it capable of estimating also the orientation of a target object. We formulate the angle estimation problem as a classification task by simply quantizing the angular range into k bins and by expanding all the C categories, managed by the object detector, into $C' = kC$ new classes. Thus, as shown in Figure 3.2, for each object instance, we can compute its real-valued angle with $\theta = \arccos(\mathbf{u}_x \cdot \mathbf{v}_x)$, as the angle between the unit vector $\mathbf{v}_x$, directed as the first edge of the corresponding OOB, and the x axis of the image $\mathbf{u}_x$. Obviously the theta angle so defined, only for illustrative intent, is limited to the range $[0, \pi]$, for this reason it is necessary to calculate it using

$$\theta = \begin{cases} \text{atan2}(v_{x_y}, v_{x_x}), & \text{if } v_{x_y} >= 0 \\ 2\pi + \text{atan2}(v_{x_y}, v_{x_x}), & \text{if } v_{x_y} < 0 \end{cases} \tag{3.1}$$

where (v_{x_y}, v_{x_x}) are the components of the $\mathbf{v}_x$ vector. The real-valued angle θ is then converted into the corresponding bin index C_θ, with $C_\theta = \lfloor \frac{\theta}{\hat{\theta}} \rceil \in \{0, ..., k\}$, where $\hat{\theta}$ is *quantization step* so that $k = \lceil \frac{2\pi}{\hat{\theta}} \rceil$. In order to build an unique formulation to obtain the final converted class we can write:

$$f_{o2u}(c_i, \theta, \hat{\theta}) = c_i k + C_\theta = c_i k + \lfloor \theta / \hat{\theta} \rceil = \hat{c}_i \tag{3.2}$$

where f_{o2u} (*i.e. o2u*: *Oriented-to-Unoriented*) is the function used to convert the original object class c_i in the expanded class $\hat{c}_i$ which encodes not only the object type but also its quantized orientation. The corresponding BB is computed simply by applying the minimum bounding box algorithm to the 4 vertices of the original OOB.

3.3.2 f_{u2o}: The Unoriented-to-Oriented Function

Assuming $y_i = \{b_i, \hat{c}_i\}$ the generic prediction of the Object Detector, where $\hat{c}_i$ is the predicted class (built with the Equation 3.2) and b_i the corresponding un-oriented bounding box, the purpose of the f_{u2o} function, as depicted in Figure 3.1, is to produce an equivalent oriented prediction $\breve{y}_i = \{\breve{b}_i, \theta_i, c_i\}$ where c_i is the original class, mapping the object of belonging only, and $\breve{b}_i$ is an oriented bounding box (when omitted in images, the angle θ_i is represented, for simplification, with the red arrow oriented as the longest axis of an OBB). This procedure can be thought as the inverse of that described in subsection 3.3.1. The function to determine the original class, c_i, and the predicted angle θ_i is pretty simple:

$$f_{u2o}(\hat{c}_i, \hat{\theta}) = \begin{cases} c_i = \lfloor \frac{\hat{c}_i}{k} \rfloor \\ \theta_i = \hat{\theta}_i \cdot (\hat{c} \bmod k) \end{cases} \tag{3.3}$$

where $\hat{\theta}$ is the same discretization step as used in the f_{o2u} counterpart. Conversely, the estimation of the OBB ($\breve{b}_i$) given the simple BB (b_i) and the corresponding angle θ_i is not a trivial problem because of the infinite number of solutions with no constraints. However, for each object in the dataset we can compute the *ratio r* of its bounding box in a nominal condition (*e.g.* when it exhibit $0°$ in an image) and use it as a constraint to reduce the complexity of the procedure. If we define with $P_{\breve{b}_i} = \{\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$ the set of OBB's 4 expressed in the reference frame F_{b_i} centered in the bounding box b_i (the clockwise order, as depicted in the last frame of Figure 3.3, has to be reliable in order to have that $\mathbf{v}_x = \frac{\mathbf{p}_1 - \mathbf{p}_0}{||\mathbf{p}_1 - \mathbf{p}_0||}$), the ratio can be computed easily as $r = \frac{\mathbf{p}_1 - \mathbf{p}_0}{\mathbf{p}_3 - \mathbf{p}_0}$. Thanks to the object aspect ratio we can execute the pipeline depicted in Figure 3.3 to build an OBB: starting from the original BB and an angle θ, we superimpose a small version of the corresponding OBB, $\breve{b}_{sm_i}$ (built only using the aspect ratio information), in the center of the BB rotating it by the provided angle; we estimate as $d = |\frac{W}{2} + p_{lm_x}|$ the distance between the leftmost vertex of the OBB, $\mathbf{p}_{lm}$, and the left edge of the original BB; we enlarge $\breve{b}_{sm_i}$ by a scaling factor s, in order to have $d = 0$. Therefore, taking the example of Figure 3.3, if $\mathbf{p}_{lm} := \mathbf{p}_3$, we can simplify the above mentioned distance condition, obtaining the new desired x coordinate of $\mathbf{p}_3$ as $\hat{p}_{3_x} = -\frac{W}{2}$. And then reformulating it as a scaling problem $\hat{p}_{3_x} = s \cdot p_{3_x}$ we get that $s = -\frac{W}{2p_{3_x}}$. The factor s can be used to generate a scaling matrix and transform all 4 vertices consistently.

As mentioned above, due to the error introduced by the Object Detector in the estimation of b_i instances, the construction of a BB from an OBB is not perfectly invertible, so the algorithm just proposed tries to minimize one of the many possible constraints (the proximity of the *left-most* point). Surely an optimization algorithm could take into account more than one constraint (e.g. the proximity of all 4 vertices) but in our case this approach is sufficiently precise and fast to test the rest of the pipeline.

3.3.3 Automatic Dataset Generation

In section 3.1 we underlined the importance to provide a smart solution to collect training data for data-driven models to be deployed in real industrial applications. For this reason, we endowed LOOP with an automated labeling tool based on video sequences. The hypotheses allowing the tool to work well are:

1. the video sequence frames a tabletop scene with the image plane as parallel as possible to the supporting surface;

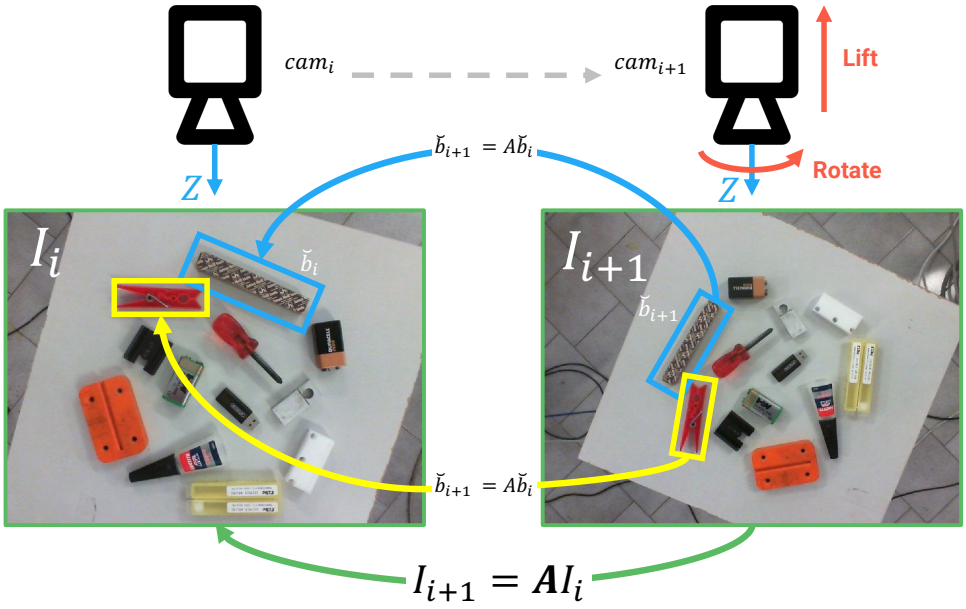


FIGURE 3.4: For each pair of consecutive frames I_i, I_{i+1} it is possible to estimate the rigid transformation $\mathbf{A}$ such that $I_{i+1} = \mathbf{A}I_i$, by means of a 2D feature matching pipeline, like *e.g.* ORB[85]. The same transformation $\mathbf{A}$ can be applied to each OOB label.

2. camera movements have to be, as much as possible, only rotational (*Rotate* around the camera z axis) and translational (*Lift* along the z axis);

3. the height of target objects has to be mostly uniform.

Figure 3.4 exemplifies the above requirements by showing two consecutive frames of a suitable video sequence. The figure shows clearly how restricted camera movements (*i.e.* lift and rotate) leads to a controlled rigid transformation $\mathbf{A}$ between the two consecutive images I_i, I_{i+1} such that $I_{i+1} = \mathbf{A}I_i$. The same rigid transformation A can be applied to each OOB $\check{b}_i$ present in the image I_i so as to obtain a new set of OOB such that $\check{b}_{i+1} = \mathbf{A}\check{b}_i$. This procedure can be repeated for each consecutive pair of images in the video sequence, it is therefore clear how the sole human intervention is to create the OOB labels in the first frame I_0. To estimate the rigid transformation $\mathbf{A}$ any 2D feature-based matching pipeline, described in section 3.2, may be used. We decided to use ORB [85], a patent-free solution, to make our software completely open-source[1].

3.4 Results

In this section we will describe both data and models used in our experiments in order to maximize reproducibility. In subsection 3.4.1 we will introduce a novel dataset, dubbed the *LOOP Dataset*, used in our quantitative and qualitative experiments. In subsection 3.4.2 we will describe how the core object detector has been trained over the LOOP Dataset in order to obtain different declensions of the more complex LOOP Detector. To prove the effectiveness of our solution, in subsection 3.4.3 we analyze the absolute LOOP's performances, while in subsection 3.4.4 we present a detailed comparison between LOOP and the state-of-the-art approaches based on handcrafted features. Finally, in subsection 3.4.5, a qualitative evaluation of our approach within a typical robotic task is provided.

FIGURE 3.5: LOOP Dataset objects grouped in *Textured* and *Untextured*.

FIGURE 3.6: In the left column: 4 samples coming from the LOOP Dataset one for each background category. The right column shows the synthetic version of each sample.

3.4.1 LOOP Dataset

Thanks to the technique examined in subsection 3.3.3, we created our an experimental –public– dataset with 12 objects equally divided into *Textured* and *Untextured* (as shown in Figure 3.5), submerged in several challenging scenes, in order to extensively compare our approach with classical ones. We collected 15 tabletop scenes, with randomly arranged objects, featuring different backgrounds: 3 scenes with *homogeneous* background; 3 scenes with *wood*; 3 scenes with *black* background and 5 scenes with an high-clutter background (several prints of *Pollock*'s painting). This increasing variability is used in order to achieve *Domain Randomization* [86] during training and very challenging scenes during the test. These 4 different scene categories are shown in Figure 3.6, where they are intentionally presented in increasing order of complexity. We collected a total of 7155 self-labeled images (strictly speaking, we manually labeled only 15 images, one per scene). Moreover, by using the *ground truth* coming from the whole dataset we also built a *Synthetic* version of it just by stitching one version of each model with the same arrangement proposed by the original dataset (right column in Figure 3.6 illustrates synthetic version of the real images in the left column). This version of the dataset is built in order to reproduce a version of the training data in which the distribution of the objects in

terms of position and orientation is identical to the real one (also the backgrounds are intentionally similarly synthetized) but the variance of objects semblance, in the images space, is purposely kept low: *i.e.* stitching the same version of the object synthetically rotated is, under our hypothesis, less effective than produce real rotated viewpoints, especially dealing with deep neural models, not to mention that even the variations of light conditions and perspective are not correctly captured by a synthetic dataset generated this way.

3.4.2 Deep Object Detector

As reference for our benchmarks we used YOLOv3 [81], a state-of-the-art Object Detector based on CNNs. We fine-tuned the YOLOv3 model, pretrained on ImageNet [53], with the LOOP approach using 13 scenes of the LOOP Dataset (about 6200 labeled images) with a 80%/20% split for training and test. The training strategy is to freeze weights of the network's feature extractor (namely, the backend), for 2 epochs training only the object detection layers (namely, the frontend), then fine-tuning the whole architecture for 50 further epochs, with a learning rate of 0.001. The two remaining scenes of the dataset (*i.e.* one with wood background, thought as a simple testbench, and one with high clutter background thought as a complex one), never used during the training phase are used to evaluate the performance of the whole system. We will call these two scenes *Simple Scene* (477 images) and *Hard Scene* (477 images) in the rest of this section.

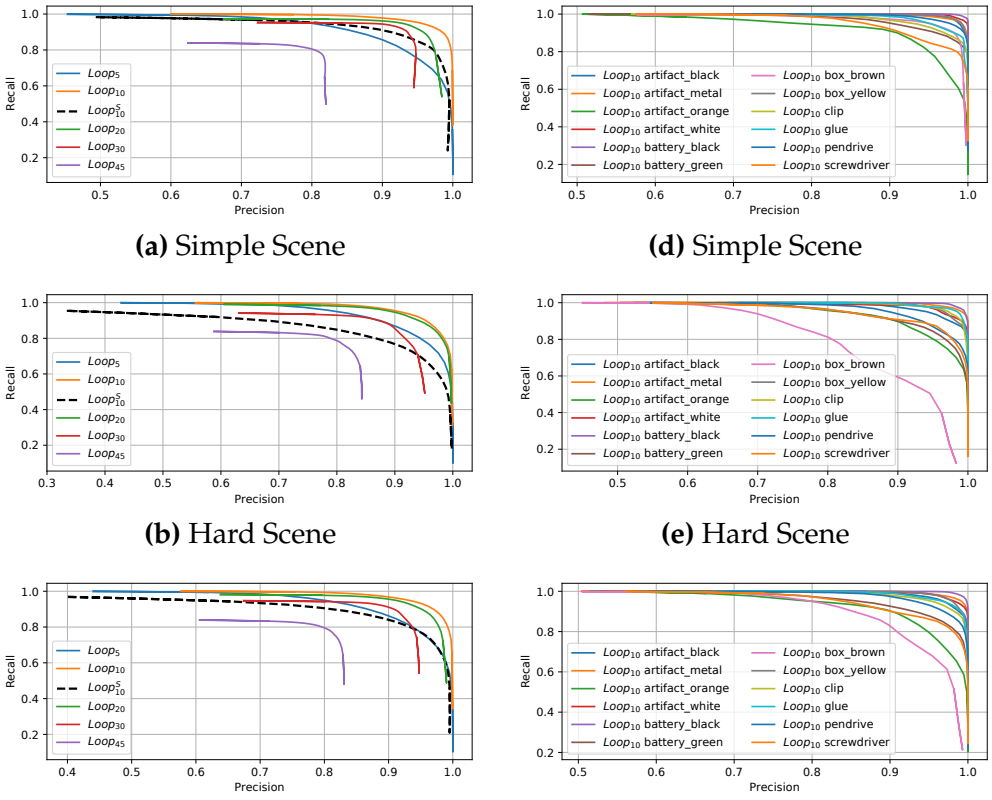
We trained several models using different $\hat{\theta}$ (*i.e.* angle discretization parameter, as described in subsection 3.3.1), thereby producing different declensions of the detector useful in understanding how the discretization factor goes to affect performances. For the sake of convenience, we will adopt the short nickname LOOP_α to identify a LOOP model trained using the discretization angle $\hat{\theta} = \alpha$ (*e.g.* LOOP_{10} identifies a model trained by quantizing the whole *angle turn* in 36 bins of 10° each). Moreover, we add the letter S to the above mentioned notation (*e.g.* LOOP^S_α) to represent the same model trained with the synthetic version of the LOOP Dataset.

3.4.3 LOOP performances

In the claim of this work we said that with the LOOP approach it is possible to build effortlessly an Object Detector, based on CNNs suitable to real industrial applications. In this section, indeed, we evaluate LOOP over the LOOP Dataset to understand if its performances are good enough to embody it into a reliable industrial system.

Figure 3.7 groups together several Precision/Recall curves obtained by varying the threshold over the confidence output of the YOLOv3 model. Figure 3.7 (a), (b) and (c) depict the precision/recall of several detectors (*i.e.* each of which trained with a different angle discretization $\hat{\theta}$) over the *Simple Scene*, *Hard Scene* and both, respectively. In the same plots also the LOOP^S_{10} model, thus trained only over synthetic data, is depicted in order to assess its performance compared to its counterparty trained on real data (*i.e.* LOOP_{10}). Table 3.1 resumes the *mean average precision* for the considered models: in this concise summary it is even clearer how deploying a synthetic dataset is significantly less effective than leveraging on real imagery, especially when it comes to very complex real situations.

From these results we can conclude that LOOP_{10} is the best version of our detector. The LOOP_5 and LOOP_{20} versions resulted slightly worse. This analysis shows,



(a) Simple Scene **(d)** Simple Scene

(b) Hard Scene **(e)** Hard Scene

(c) Overall **(f)** Overall

FIGURE 3.7: Precision/Recall curves for the LOOP approach. These plots have been produces by varying the threshold of the predictions *confidence* between 0.0 and 1.0 with a step of 0.05. The first column compares various discretization angles. The second column shows performances of the best version, LOOP_{10}, for each single object in the dataset.

Model	Simple	Hard	Overall
$LOOP_5$	0.96	0.96	0.96
$LOOP_{10}$	**0.99**	**0.98**	**0.99**
$LOOP^S_{10}$	0.95	0.86	0.92
$LOOP_{20}$	0.95	0.97	0.96
$LOOP_{30}$	0.90	0.88	0.89
$LOOP_{45}$	0.69	0.70	0.69

TABLE 3.1: The mean average precision (*mAP*) computed for each model, across all objects, for the *Simple Scene*, *Hard Scene* and both.

clearly, how the angle discretization parameter $\hat{\theta}$ affects performances in both direction: too small a value may cause a dramatic increase of detector categories C' (*e.g.* with $\hat{\theta} = 5$ we have $C' = \frac{360}{5}12 = 864$); on the other hand, if $\hat{\theta}$ is too high, the angle prediction is subject to a large discretization error producing low IoU scores. Moreover, training on synthetic data only ($LOOP^S_{10}$), although acceptable, is worse than training on real data. With regard to single objects precision/recall, as will be described in more detail later, symmetric objects are – unsurprisingly – somewhat confusing the algorithm.

3.4.4 Hand-crafted features vs Deep Learning

We used our LOOP Dataset to test the LOOP approach compared with the state-of-the-art algorithms based on hand-crafted features. The first competitor is, certainly, SIFT [73], one of the most used feature-based approach for *Textured* objects detection. For the *Untextured* counterpart, we chose BOLD [75], which uses highly repeatable geometric primitives as 2D features in order to perform object detection also with monochrome targets. Both methods are implemented in the same object detection pipeline as described in [75]: 1) key-points detection; 2) key-point description; 3) features correspondences validated through Generalized Hough Transform (as described in [73])and 4) Pose Estimation. The Pose Estimation in this case is modified in order to obtain a Rigid Transform instead of a Full Affine (homography). In this way we are sure that both detectors will yield predictions similar to $\check{y}_i = \{\check{b}_i, \theta_i, c_i\}$ without producing distorted bounding boxes, as it is the case of homographies.

We compare the output of the previous two pipelines with the output directly obtained with the LOOP approach. We measured the performances with the classical precision/recall metric computed taking into account the *Intersection Over Union* (IoU) of the predicted OBB: each detection, of a generic algorithm, is counted as *True Positive* if its IoU is > 0.5 compared with the corresponding ground truth OBB. With this metric we can analyse: Precision, Recall, FScore and Average IoU for each algorithm. We introduce also an additional term called *Oriented Intersection Over Union* (OIoU) which measures the classical IoU commesurate to the effectiveness of the algorithm in predicting the correct angle. Formally $\text{OIoU} = \text{IoU} \times \max\left(\frac{\mathbf{v}_x \cdot \hat{\mathbf{v}}_x}{\|\mathbf{v}_x\|\|\hat{\mathbf{v}}_x\|}, 0\right)$, so the Oriented IoU is inversely proportional to the angle between the two $\mathbf{v}_x$ of the ground truth and the predicted OBB; it is forced to be between 0 and 1, with the *max* operator, in order to discard a priori opposed angles. The OIoU term is introduced to measure the capability of each algorithm in distinguishing quite symmetric objects: in the LOOP Dataset, for instance, the two objects *artifact_orange* and

box_brown seems very symmetrical even thought they are not. In such cases, if the angle prediction is completely wrong, the IoU may be high but the OIoU is very low.

Under the assumption that, as deduced in subsection 3.4.3, the model with the best trade-off between precision and recall is $LOOP_{10}$, we used the latter to compete with the state-of-the-art. In Table 3.2 and Table 3.3 several comprehensive results are shown. The first table compares SIFT and BOLD with $LOOP_{10}$ and $LOOP^{S}_{10}$ dealing with the *Simple Scene*, *i.e.* the scene with a low clutter background. The second table, instead, deals with the *Hard Scene* containing an highly cluttered background. Both tables contain performances dealing with each object, a summary for Textured and Untextured objects and an Overall index. As vouched by these experimental results, LOOP outperforms both SIFT and BOLD in both scenarios: our approach is able to cope with both Textured and Untextured objects seamlessly and is very robust to the high level of clutter in the *Hard Scene*. On the other hand, as pointed out in Table 3.2, the OIoU index shows a slight fall of LOOP when dealing with symmetric objects (*e.g.* the *box_brown* OIoU is 0). This conceptual problem can be easily resolved treating symmetrical objects differently during the angle discretization process, described in subsection 3.3.1, with a formulation similar to the one introduced in [65]. A qualitative evaluation is present in Figure 3.9 featuring a real output of the LOOP detector on two samples randomly picked between the Simple and the Hard scenes subsets.

Index	Precision				Recall				FScore				IOU				OIOU			
Method	Sift	Bold	$LOOP_{10}$	$LOOP^{S}_{10}$	Sift	Bold	$LOOP_{10}$	$LOOP^{S}_{10}$	Sift	Bold	$LOOP_{10}$	$LOOP^{S}_{10}$	Sift	Bold	$LOOP_{10}$	$LOOP^{S}_{10}$	Sift	Bold	$LOOP_{10}$	$LOOP^{S}_{10}$
artifact_black	0.50	0.84	**0.99**	0.97	0.00	0.81	0.96	**0.98**	0.00	0.82	**0.97**	0.97	0.75	0.77	**0.85**	0.84	0.75	0.75	**0.85**	0.84
artifact_metal	0.00	0.00	**0.97**	0.95	0.00	0.00	**0.98**	0.86	0.00	0.00	**0.97**	0.90	0.00	0.00	**0.77**	0.77	0.00	0.00	**0.77**	0.77
artifact_orange	0.54	**0.96**	0.91	0.78	0.12	**0.96**	0.91	0.88	0.20	**0.96**	0.91	0.83	0.71	**0.86**	0.86	0.86	0.62	0.43	**0.78**	0.37
artifact_white	0.62	0.00	**0.97**	0.90	0.85	0.00	**1.00**	0.98	0.72	0.00	**0.98**	0.94	0.78	0.00	**0.84**	0.80	0.78	0.00	**0.84**	0.79
clip	0.78	0.60	**0.95**	0.91	0.63	0.60	**0.93**	0.92	0.70	0.60	**0.94**	0.91	0.76	0.76	**0.81**	0.75	0.76	0.74	**0.81**	0.74
screwdriver	0.76	0.58	0.95	**0.97**	0.05	0.58	0.85	**0.89**	0.10	0.58	0.89	**0.93**	0.71	0.73	**0.81**	0.79	0.71	0.72	**0.81**	0.79
Untextured	0.53	0.50	**0.95**	0.91	0.28	0.49	**0.94**	0.92	0.29	0.49	**0.95**	0.91	0.62	0.52	**0.82**	0.80	0.60	0.44	**0.81**	0.72
battery_black	0.48	0.82	**0.98**	0.94	0.81	0.82	1.00	**1.00**	0.60	0.82	**0.99**	0.97	0.76	0.71	**0.84**	0.83	0.75	0.68	**0.84**	0.83
battery_green	0.65	0.44	**0.93**	0.90	0.83	0.43	**0.93**	0.91	0.73	0.43	**0.93**	0.90	0.69	0.70	0.85	**0.85**	0.69	0.59	**0.82**	0.79
box_brown	0.65	0.68	**0.97**	0.87	0.77	0.68	0.92	**0.95**	0.71	0.68	**0.95**	0.91	**0.80**	0.75	0.74	0.73	**0.80**	0.65	0.00	0.02
box_yellow	0.49	**1.00**	0.98	0.98	1.00	**1.00**	0.98	0.59	0.66	**1.00**	0.98	0.73	0.82	0.81	**0.86**	0.80	0.82	0.81	**0.86**	0.79
glue	0.37	**0.99**	0.97	0.95	**1.00**	0.99	0.91	0.97	0.54	**0.99**	0.94	0.96	0.80	0.81	0.82	**0.83**	0.80	0.80	0.82	**0.83**
pendrive	0.78	0.91	**0.98**	0.92	0.36	0.91	**0.94**	0.85	0.49	0.91	**0.96**	0.89	0.75	0.74	**0.76**	0.73	0.74	0.73	**0.76**	0.71
Textured	0.58	0.79	**0.97**	0.92	0.75	0.78	**0.94**	0.86	0.61	0.78	**0.95**	0.88	0.77	0.76	**0.81**	0.79	**0.77**	0.71	0.67	0.64
global	0.54	0.72	**0.96**	0.91	0.54	0.65	**0.94**	0.90	0.54	0.68	**0.95**	0.91	0.77	0.77	**0.82**	0.80	**0.77**	0.70	0.75	0.69

TABLE 3.2: Performances of our approach, compared to SIFT[73] and BOLD[75], in the *Simple Scene* scenario (boldface text highlights the best score in the related pane). The overall Precision/Recall index of LOOP is about 96%/94% showing its flexibility in general purpose real applications.

Index	Precision				Recall				FScore				IOU				OIOU			
Method	Sift	Bold	$LOOP_{10}$	$LOOP^{S}_{10}$	Sift	Bold	$LOOP_{10}$	$LOOP^{S}_{10}$	Sift	Bold	$LOOP_{10}$	$LOOP^{S}_{10}$	Sift	Bold	$LOOP_{10}$	$LOOP^{S}_{10}$	Sift	Bold	$LOOP_{10}$	$LOOP^{S}_{10}$
artifact_black	0.00	0.01	**0.96**	0.90	0.00	0.01	**0.94**	0.90	0.00	0.01	**0.95**	0.90	0.00	0.64	**0.80**	0.75	0.00	0.05	**0.80**	0.74
artifact_metal	0.43	0.00	**0.97**	0.83	0.01	0.00	**0.98**	0.93	0.02	0.00	**0.97**	0.87	0.70	0.00	**0.82**	0.76	0.70	0.00	**0.82**	0.76
artifact_orange	0.75	0.00	**0.91**	0.78	0.41	0.00	0.88	**0.92**	0.53	0.00	**0.89**	0.85	0.85	0.00	**0.87**	0.86	**0.84**	0.00	0.82	0.68
artifact_white	0.73	0.00	**0.96**	0.80	0.10	0.00	0.96	**0.98**	0.18	0.00	**0.96**	0.88	0.68	0.00	**0.80**	0.80	0.68	0.00	**0.79**	0.76
clip	0.00	0.40	**0.96**	0.91	0.00	0.40	**0.97**	0.87	0.00	0.40	**0.97**	0.89	0.00	0.71	0.79	**0.80**	0.00	0.63	**0.79**	0.78
screwdriver	0.00	0.11	0.94	**0.94**	0.00	0.11	**0.90**	0.48	0.00	0.11	**0.92**	0.63	0.00	0.75	**0.76**	0.75	0.00	0.75	**0.76**	0.74
Untextured	0.32	0.09	**0.95**	0.86	0.09	0.09	**0.94**	0.85	0.12	0.09	**0.94**	0.84	0.37	0.35	**0.81**	0.79	0.37	0.24	**0.80**	0.74
battery_black	0.59	0.91	**0.98**	0.91	0.58	0.91	**0.99**	0.97	0.58	0.91	**0.98**	0.94	0.74	0.61	**0.84**	0.81	0.74	0.24	**0.84**	0.81
battery_green	0.67	0.04	**0.94**	0.84	**0.90**	0.04	0.87	0.50	0.76	0.04	**0.90**	0.63	0.68	0.62	**0.80**	0.75	0.68	0.54	**0.75**	0.71
box_brown	0.37	0.24	**0.78**	0.74	**0.99**	0.24	0.84	0.46	0.54	0.24	**0.81**	0.57	**0.80**	0.70	0.74	0.69	**0.80**	0.69	0.51	0.23
box_yellow	0.47	**1.00**	0.96	0.96	0.82	**1.00**	0.96	0.97	0.60	**1.00**	0.96	0.97	**0.85**	0.84	0.84	0.83	**0.85**	0.84	0.84	0.83
glue	0.59	0.94	**0.99**	0.94	0.81	0.94	**0.95**	0.79	0.68	0.94	**0.97**	0.86	0.78	0.79	**0.84**	0.79	0.78	0.79	**0.84**	0.79
pendrive	0.88	0.81	**0.96**	0.90	0.09	0.81	**0.85**	0.81	0.17	0.81	**0.90**	0.85	0.64	0.63	**0.78**	0.73	0.64	0.55	**0.78**	0.71
Textured	0.58	0.58	**0.93**	0.88	0.67	0.57	**0.90**	0.72	0.53	0.57	**0.91**	0.78	0.75	0.72	**0.80**	0.76	**0.75**	0.67	0.75	0.67
global	0.53	0.42	**0.94**	0.87	0.39	0.37	**0.92**	0.80	0.45	0.40	**0.93**	0.83	0.77	0.72	**0.81**	0.78	0.77	0.62	**0.78**	0.73

TABLE 3.3: Performances of our approach, compared to SIFT[73] and BOLD[75], in the *Hard Scene* scenario (boldface text highlights the best score in the related pane). Here, the overall Precision/Recall index of LOOP is about 94%/92% showing its robustness against high clutter backgrounds, a typical situation in real industrial environments.

3.4.5 Real Robotic application

As a qualitative evaluation of our approach we designed a proof-of-concept pick&place robotic application based on the outcome of the LOOP detector. Note this application is similar to the wire-terminal picking in section 2.5. In fact, we use almost the same experimental setup presented in section 2.4. In particular, we use an industrial robotic arm (COMAU smart six) with a parallel gripper as end effector and an eye-on-hand camera on board. The camera mounted in this way can be thought as the secondary end-effector and than can be arbitrarily moved with high precision. The image plane of the camera is kept parallel to a table-top scene with randomly arranged objects belonging to the LOOP Dataset (same camera-table configuration seen in subsection 3.3.3). The distance between camera and the table plane is known. Given that the output of the pipeline is a set of oriented prediction $\check{y}_i = \{\check{b}_i = \{x_i, y_i, w_i, h_i\}, \theta_i, c_i\}$ we exploit their position (x_i, y_i) and orientation (θ_i) to build a simple control scheme, for robot guidance, in such a way as to move the camera in order to align a target object with the center of camera viewpoint, oriented as the canonical x axis of the image. A proportional-only control scheme is used, thus, to minimize $\mathbf{e}_t$ and e_θ, the translational and rotational error respectively $\mathbf{e}_t = \{x_i - c_x, y_i - c_y\}$ and $e_\theta = -\text{sign}(\sin(\theta_i))(\cos(\theta_i) - 1)$, with (c_x, c_y) the center of the image. The control scheme will produce a linear velocity $\mathbf{v} = K_{P_t}\mathbf{e}_v$ and an angular velocity $\omega = K_{P_\theta} e_\theta$ to the end effector, *i.e.* the camera reference frame (K_{P_t} and K_{P_θ} are tunable proportional gains). The $\text{sign}(\cdot)$ function is simply the *sign* function with $\text{sign}(0) = 1$ to avoid singularities. The above simplified robot guidance scheme is used to lead the robot in a easier condition suitable to estimate the 3D pose of the object because: knowing the extrinsics parameters of the mounted camera and the height of the table w.r.t. the robot base, it is trivial to know the 3D coordinates of an object centered in the camera viewpoint. Figure 3.8 exemplifies the described procedure by presenting a real execution of it. In the supplementary material several runs of experiment are shown from the on-board and off-board cameras. We have accomplished this task on 20 scenes with completely unseen backgrounds. The overall success rate of the final grasp is 100% for each object, except for *artifcat_orange* and *box_brown*, which instead scored 60% and 50% respectively. Hance, the success rate for the entire dataset is 92.5%. As expected from the conclusions of subsection 3.4.4, the grasp for highly symmetrical objects (like *artifcat_orange* and *box_brown*) becomes very challenging, due to the difficulty of estimating unambiguous orientation for these samples. In these cases, the indecision of the detector combined with the stateless nature of a CNN-based Object Detector (*i.e.* there is no online tracking method like in [87], where a Recurrent Neural Network is used in order to achieve a continuous estimate of the pose) leads to a detrimental oscillation of the output in the control module. Then, we corrected the model by modifying the rotational error in such a way as to treat objects as symmetric, with

$$\begin{cases} \mathbf{e}_t = \{x_i - c_x, y_i - c_y\} \\ e_\theta^{\text{Sym}} = \text{sign}(\tan(\theta))(|\sin(\theta)|) \end{cases} . \tag{3.4}$$

Accordingly, the robot guidance scheme leads the camera to move towards the nearest horizontal configuration of the target object (0° or 180° indifferently), reaching 100% of success rate also for each object (the control schemes were tested again over 20 scenes).

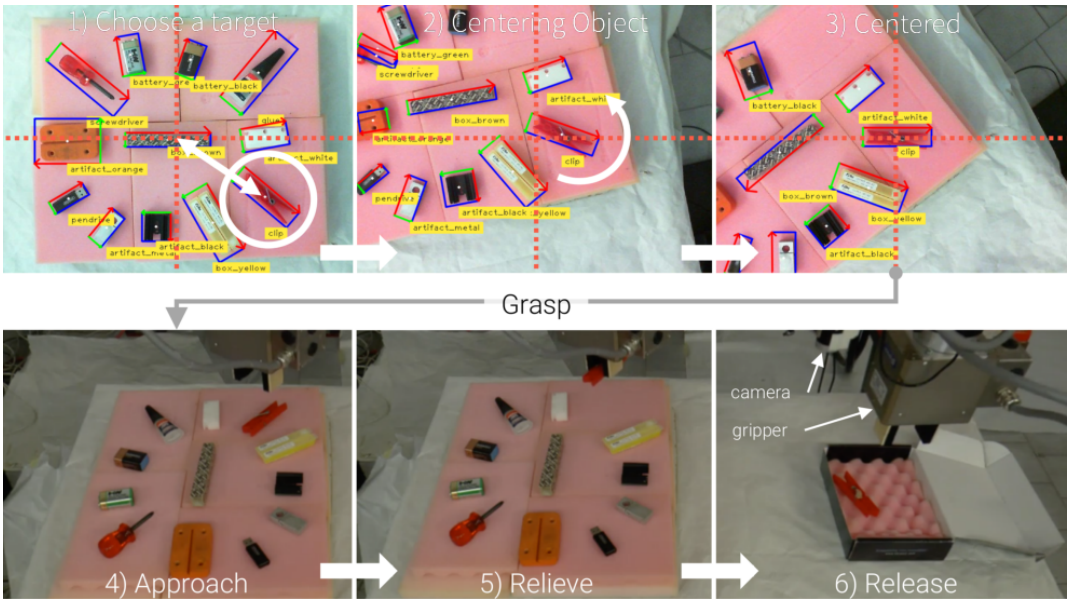


FIGURE 3.8: An exemplification of the pick-and-place system designed using LOOP as a module of the control scheme. The first row depicts frames coming from the on-board camera, with superimposed predictions, showing the robot guidance control phases. The second row shows images coming from an off-board camera framing the grasp sequence downstream of the alignment procedure.

3.5 Conclusions

In this chapter we proposed an extension of classical CNN-based Object Detectors able to produce Oriented Bounding Boxes suitable for the 3-DoF pose estimation task. We provided our detector with a simplified procedure to gather a huge amount of training data in the field, with trifling human intervention. With this work we extend the solution pretended in subsection 2.5.1 while we provide a possible improvement also the pose estimation of the wire-terminals studied in subsection 2.5.2 and subsection 2.5.4. More in general we can state that, this work shows how to effectively use Deep Learning in a real industrial setting, exploiting a neural network as a module of a more complex control scheme.

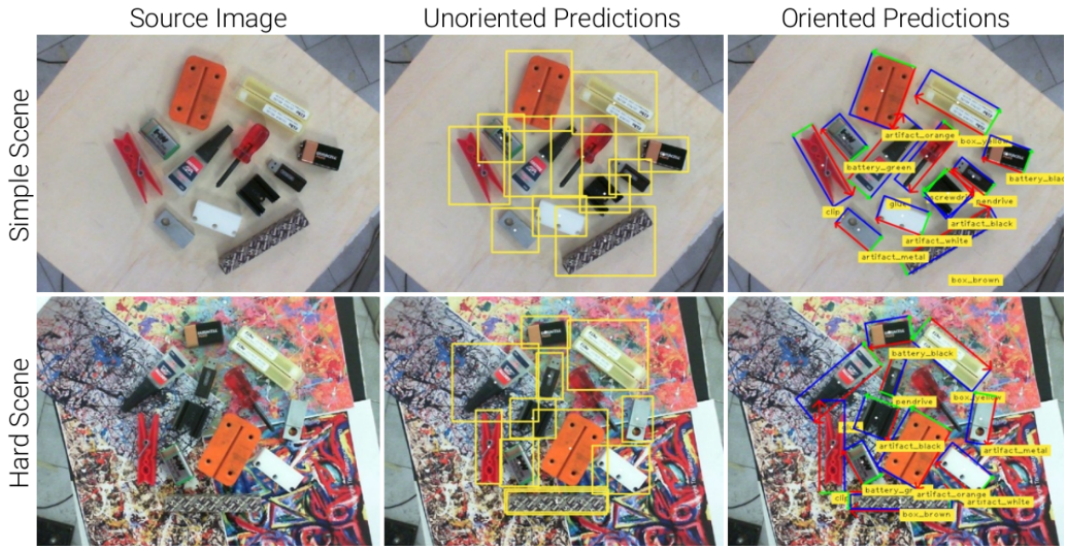


FIGURE 3.9: This picture shows the real output of our software for the simple and hard scene, respectively. The middle column shows Unoriented Predictions (BB) coming from the generic object detector, while the right column shows the derived Oriented Predictions (OBB).

Chapter 4

Insertion of a DLO in a Hole using Tactile Data

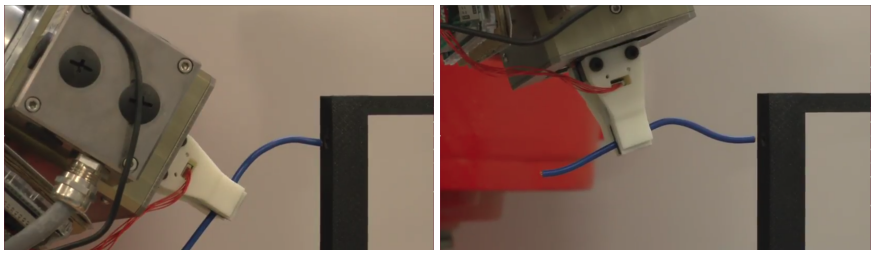
In subsection 2.5.5 we trained a regressor to infer, from tactile sensor data, the contact forces exerted on a wire terminal. However, the pipeline presented in chapter 2 uses this regressor only to monitor the success of the wire-terminal insertion task, and possibly replan the operation. Moreover, this procedure relies on the simplifying assumption of grasping the wire sufficiently close to the terminal to be considered always straight and stiff. Thus, the insertion task of subsection 2.5.5 is not very different from a peg-in-hole task in open-loop.

In this chapter, we aim to improve the this component of the system by developing a closed-loop strategy to guide the insertion of the electrical wire into a hole by analyzing the feedback coming from a tactile sensor. We also tackle issues related to the deformability of the cable, empowering the insertion of longer and curved wire segments too.

4.1 Introduction

Traditionally, in an robotic assembly systems, the interaction between the end-effector and the environment during task is measured by means of a Force/Torque (F/T) sensor placed between the robot's wrist and tool. This F/T configuration impedes to directly measure the forces/torques exerted on the tool-tip, not to mention the fact that measures will be distorted by the inertial effects acting on the tool itself. Moreover, in situation like the one described in section 2.4, where an additional multi-joint gripper like the one in Figure 2.4 is mounted downstream of the force sensor, non-static system dynamics prevent robust control based on force/torque feedback. Thus, as already proposed in chapter 2, we replace the classic wrist-mounted F/T sensor with by a finger-mounted tactile sensor which, by exploiting a Recurrent Neural Network (RNN), has been trained to emulate the more complex F/T output. In this way, as discussed in [8], we are able to estimate all the forces acting on the cable (not only the classic ones of a tactile sensor orthogonal to its contact surface), with a sensor placed directly in contact with it and therefore free of further external disturbances.

The remainder of this chapter is structured as follows. section 4.2, reports an overview of previous works in this field. section 4.3 describes the sub-system used to estimate external forces exerted on the wire through the tactile sensor. section 4.4 describes conceptually the insertion algorithm, while section 4.5 provides the control equations. Finally, in section 4.6, in order to validate the proposed approach several experiments will be provided.



(A) Configuration 1 (C1). (B) Configuration 2 (C2).

FIGURE 4.1: Two different (most likely) configurations of an elasto-plastic wire in real scenarios. On the left the wire features a simple curvature. On the right, instead, an s-shaped configuration.

4.2 Related Works

For an autonomous system, the main problem in DLOs manipulation is dealing with uncertainty about their shape. The natural approach is the use of vision systems in order to ameliorate these uncertainties. Inaba *et al.*[88] developed an hand-eye system to insert a rope into a hole. In their work, the insertion was performed by measuring the rope's tip center using stereo vision, and by computing the relative position between tip and target hole. An alternative approach is presented in [89] where a desired trajectory equivalent to the curve formed by the DLO is defined. When the clearance between the DLO and the hole detected by vision is tight, then the position and orientation of the tool is adjusted. In [90] the authors propose a method to estimates forces acting on a flexible wire by measuring its shape through a 3D reconstruction. More robust results were obtained fusing force and visual data together. In these works a model of the wire is computed and updated in real-time so as to match the observations of the real cable. In [91], for instance, the authors propose a method for calculating the shape of a filiform object by minimizing the energy of its configuration. Then, using the above mentioned technique, a method for inserting a flexible wire is developed in [92], where the amount of plastic deformation is estimated by stereo vision and an F/T sensor. Unfortunately, in many real applications, such as the one under examination, the reduced space of the system does not allow the use of a vision system with a not negligible encumbrance. To overcome the gripper's inertial effects, that clearly plays a fundamental role in every force control system, the robotics community worked for a long time on several solutions [93], [94], [95], [96]. The main idea developed is to estimate the force/torque measures corresponding to the contact of the tool-tip with an object by compensating the inertial forces and moments caused by the tool by means of wrist-mounted inertial sensors. In our work, as stated before, we avoid the inertia problem by replacing the F/T sensor with a finger-mounted tactile sensor directly in contact with the object to be manipulated. Control strategies based on force/torque measures or prior knowledge on the DLO's dynamic model are therefore developed. In robotic surgery, an automated electrode array insertion, for cochlear implant, was developed in [97]. This application domain, as stated in the work, does not allow 3D vision systems, thus an hybrid position-admittance control is proposed, where the insertion path planner is modified by introducing in-vivo force measurements in order to reduce sensitivity to misalignment errors.

At the best of our knowledge, our system is the only one able to completely replace the force sensor in the control scheme, with a tactile sensor whose capabilities

have been augmented by means of an RNN used to regress the three components of the force acting on the cable, even that tangent to the plane of the sensor itself, all without exploiting any visual feedback.

4.3 Tactile Feedback

The estimation of position, orientation and buckle of the grasped object is very useful for executing an insertion task. Unfortunately, in many real applications, e.g. wiring tasks, the narrow spaces available in the workspace may not be enough for using a vision system alone and sometimes is not even feasible. In these situations, the controller relies only on a force/torque feedback. In this work, a tactile sensor with 16 optoelectronic taxels and a flat deformable layer has been specifically designed for wires grasping [98]. Tactile data are often interconnected and noisy. Machine learning approaches have already extensively been used to relate such complex data to object classes [45] and grasping stability [99], for instance. The tactile data have been used by three regressors able to map the 16 signals into a three-dimensional vector which represents an estimation of the force vector $\boldsymbol{f}(t) \in \mathbb{R}^3$ detected on the finger. We define $\boldsymbol{f}$ with respect to the current tool-frame {TL}, namely the frame attached to the robot end-effector, which has origin $\boldsymbol{p}_{tl}$ centred in the gripper finger and axes defined by the unit vectors $\bar{\boldsymbol{x}}_{tl}$, $\bar{\boldsymbol{y}}_{tl}$ and $\bar{\boldsymbol{z}}_{tl}$. This force estimate can be later used to describe the interaction of the grasped object with the environment. In order to evaluate this procedure, a simple controller has been developed to perform the insertion of a flexible electric wire in an hole, as described in section 4.5.

4.3.1 Data Gathering

Since we want the networks to map the 16-dimensional tactile signal into a unbiased and undisturbed 3-dimensional force signal, we used as ground truth the signal produced by a strain gauge F/T sensor, mounted between the robot wrist and the two-fingers gripper (with axes parallel to the tool-frame {TL} axes), cleared out by the biases (e.g. gravity) and smoothed with a moving average filter. Moreover, the data gathering phase has been designed ad hoc to avoid the inertial/gravity distortion effects and other disturbance such as undesired points of contact with the environment (apart from the contact with the grasped wire). In each iteration of this phase: 1) the robot grasps a straight wire from a well known position, with a pose displacement changing in every iteration; 2) the tactile sensor is reset to remove the offsets of the specific grasp and read only the deformation on the wire; 3) the robot presses the wire terminal on a board along its three Cartesian axes $\bar{\boldsymbol{x}}_{wt}$, $\bar{\boldsymbol{y}}_{wt}$ and $\bar{\boldsymbol{z}}_{wt}$, performing 6 distinct moves, one for each axis direction. The wire terminal axes, here defined only for the data gathering phase, are always such that the plane $\{\bar{\boldsymbol{y}}_{wt}, \bar{\boldsymbol{z}}_{wt}\}$ is parallel to the gripper plane $\{\bar{\boldsymbol{y}}_{tl}, \bar{\boldsymbol{z}}_{tl}\}$. During these moves the sensors data are collected for the training. Each move is linear along one wire terminal axis, in order to acquire only one force component exerted on the wire decoupled from the others two, as depicted in Figure 4.2. The end-effector is moved slow enough to consider the inertial effects negligible while the gravity effects are compensate and the force measures transformed in the wire terminal frame. This procedure is iterated on 30 different grasp and each iteration last 40 s. The data are sampled with period 25 ms, hence we collected 48000 samples.

4.3.2 Tactile-Force Regression

The three regressors implemented in this work are three Recurrent Neural Networks (RNNs) with Long-Short Term Memory (LSTM) cells. RNNs are basically designed to process time-series data, and they have achieved good performance in applications such as speech and text recognition [100]. All RNNs have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, i.e. a single layer with an an activation such as hyperbolic tangent. In theory, these networks are absolutely capable of handling short and long-term dependencies in the data sequences. In practice, they have a tendency to suffer from vanishing or exploding gradients: back-propagated through time error signals either shrink rapidly, or grow out of bounds [101]. To overcome this problem, LSTM [102] was proposed, which incorporates memory units and gate functions.

Each RNN built in our system consists of a sequence of 2 LSTM layers (with 100 cells each), and a densely-connected Neural Network (3 hidden layers with 16, 8, 4 neurons respectively and with hyperbolic tangent activation function) to process the output data. The networks are fed with a tactile data sequence of length 5 (read window size), where each sample is taken with a period 25 ms and has 16 features (number of sensor cells). The sequences themselves are sliding windows and hence shift by 1 step each time, causing a constant overlap with the prior windows.

The predictions of the RNNs, on test data, are compared in Figure 4.3 with the F/T sensor reads on the 3 axes and the prediction of a Artificial Neural Networks (ANN) (1 hidden layer with 16 neurons and hyperbolic tangent activation function). The hyper-parameter of this ANN are found by means of a grid search procedure. As already visible from the figure, the RNN can efficiently decoupled the force components while the ANN shows in each axis the effects of the other two axes too. Also the Mean Average Errors (MAE) obtained on the test data with the RNN, i.e. 0.0384, 0.0789, 0.0718 for x, y and z axis respectively, are significantly smaller than those obtained with the ANN, i.e. 0.3755, 0.3247, 0.3033 on the same axes.

The forces estimated by the RNN along each axis are cut by a threshold-based function. Let $\hat{f}_i$ be the raw data produced by the regressor for the i-th axis and f_i^{thr} the correspondent threshold, we cut this through the function $f_i = f_i(\hat{f}_i)$, that is an identity for $\left|\hat{f}_i\right| > f_i^{\text{thr}}$ and null otherwise. We also assume $\boldsymbol{f}(t) = \mathbf{0}, \forall t < 0$.

4.4 Insertion Task

Inserting a wire which has been plastic deform into a hole can be difficult even for a human being for the first time. This is especially true when a large amount of friction is generated between the edges of the hole and the wire. In fact such a friction complicates the insertion task because the wire is easily bent by external forces. In flexible peg-in-hole tasks, it can be assumed that a part is deformed, buckled or bent, only by the axial or lateral forces. In most real applications, the wire deformation is locally contained, thus we can consider only deformations modeled by a polynomial up to the third order absolutely monotonic, Figure 4.1.

The insertion task starts by simply moving the wire tip in the direction of the hole axis, therefore we need only to know the tip position and orientation. If the manipulator senses a larger resistance force than a given value during the insertion, the position and orientation of the end-effector is adjusted to decrease the resistance force.

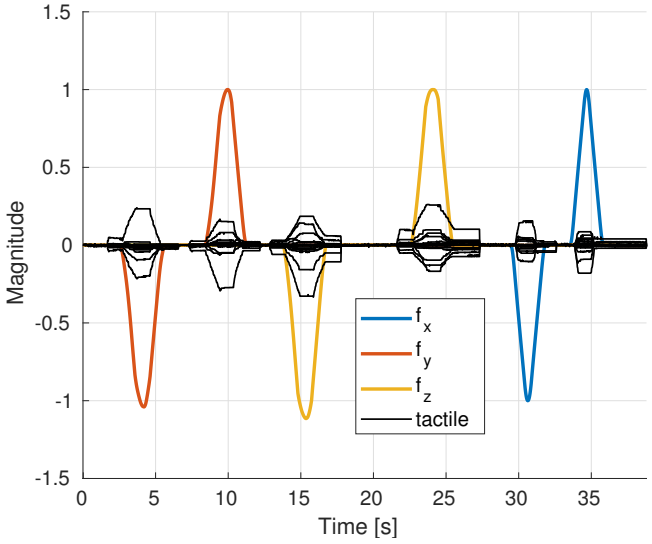
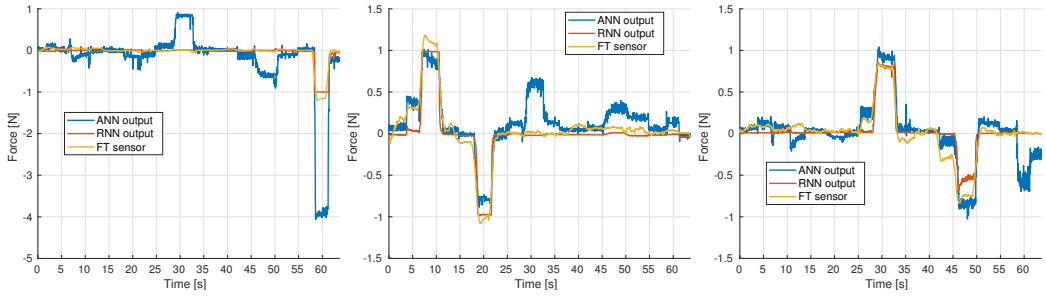


FIGURE 4.2: Training data of one single grasp. The tactile input data consists of 16 signals, one for each cell of tactile sensor. The ground truths are 3 signals, one for each axis for the force sensor.

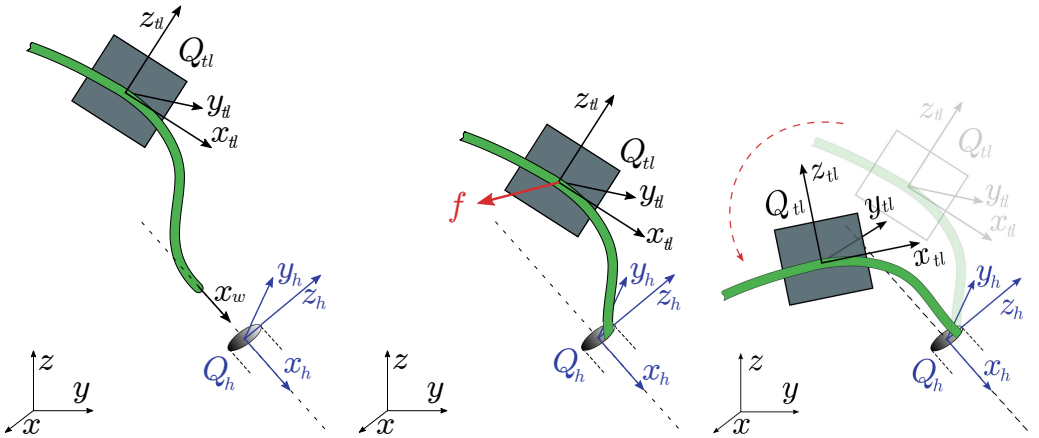


(A) X-axis. (B) Y-axis. (C) Z-axis.

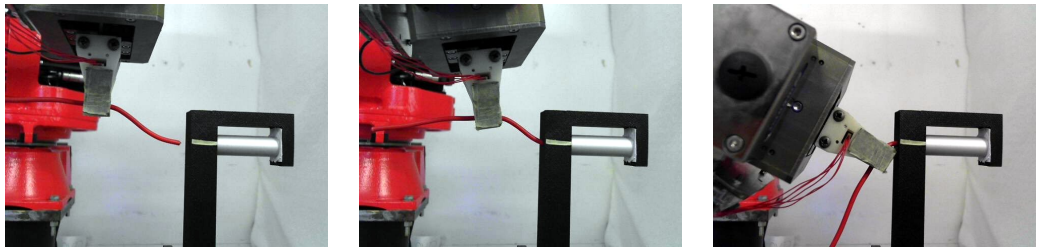
FIGURE 4.3: Here we compare the force signals given by the FT-sensor and the predictions yield by the ANN end the RNN on the same test data. The data are acquired exerting three distinct forces along the three F/T sensor axes.

To perform the insertion we assume that the tip of the wire is initially aligned with hole's longitudinal axis $\bar{x}_h$. The hole-frame {H}, namely the frame of the hole where we want to insert the DLO, has origin in p_h. The frame axis $\bar{x}_h$ is pointing inside the hole and it is aligned with the hole's axis, while the axes $\bar{y}_h$ and $\bar{z}_h$ lie in a plane parallel to the hole surface. We also assume the hole reference frame to be known and fixed. The controller leads the robot toward the hole while it adjust the gripper pose in order to reduce the estimated forces (if greater than some thresholds). We can isolate the following three control actions:

- The **forward translation** control leads the tool in a linear motion along the hole axis $\bar{x}_h$ with velocity inversely proportional to the norm of the force. This implies that the forward motion is slowed down or even stopped when the forces along the three axes are over the threshold, preventing the wire curling.
- **backward translation** control is a linear motion proportional to the force along $\bar{x}_h$. This could lead to a backward movement along $\bar{x}_h$ facilitating the rotational correction.



(a) Initial condition. (b) Wire-hole contact. (c) Tool-pose correction.



(d) Initial condition. (e) Wire-hole contact. (f) Tool-pose correction.

FIGURE 4.4: Starting from the configuration depicted in (a), (d), the tool moves straight forward along the $\bar{x}_h$ axis. The contact between the wire hump and the hole edge is detected by tactile sensor (b), (e). The controller lead the system to minimize the force on the wire combining a translations and a rotation on of the tool (c), (f).

- The **angular regulation** control combines the rotation about the axis $\bar{y}_h$ and $\bar{z}_h$. Such rotations intend to minimize the force component on the two axes.

The task is considered accomplished when the force f reaches the equilibrium contact force f^* and the distance between tool and hole reaches a desired value Δ_p^*, which means that the tool enters in a neighbourhood of the hole approaching point.

4.5 Controller Design

Let us introduce the error state vector with the following equation:

$$\boldsymbol{e}(t) = \begin{bmatrix} e_{\rho_r}(t) & e_{\theta_y}(t) & e_{\theta_z}(t) & e_d(t) \end{bmatrix}^\top \in \mathbb{R}^4 \tag{4.1}$$

whose components are error functions defined as

$$\begin{aligned} e_{\rho_r}(t) &= \left| \min\left(0, \bar{\boldsymbol{x}}_h^\top R_{tl}(t) \boldsymbol{f}(t)\right) \right| - f_r^* \\ e_{\theta_y}(t) &= R_{tl}(t) \bar{\boldsymbol{z}}_{tl}(t)^\top \boldsymbol{f}(t) \\ e_{\theta_z}(t) &= R_{tl}(t) \bar{\boldsymbol{y}}_{tl}(t)^\top \boldsymbol{f}(t) \\ e_d(t) &= \|\boldsymbol{p}_h - \boldsymbol{p}_{tl}(t)\| - \Delta_p^* \end{aligned}$$

where the first element of e_{ρ_r} is the negative part of the force projected on the hole axis $\bar{\boldsymbol{x}}_h$, $R_{tl}(t)$ is the rotation matrix between the hole and the tool frames and $f_r^* \in \mathbb{R}^+$ is the desired contact force (along $\bar{\boldsymbol{x}}_h$); e_{θ_y} and e_{θ_z} are the force errors along the axes $\bar{\boldsymbol{z}}_{tl}$ and $\bar{\boldsymbol{y}}_{tl}$ respectively; e_d is the distance between tool $\boldsymbol{p}_{tl}$ and hole $\boldsymbol{p}_h$ with an offset $\Delta_p^* \in \mathbb{R}^+$, corresponding to the desired distance.

In the next section we develop a controller that makes the error state Equation 4.1 converge to the origin of $\mathbb{R}^4$. The target pose, Q_{tl}^*, is updated through the control matrix M, that is a transformation with respect to (wrt) the base frame {B}.

4.5.1 Translation Control

We decompose the transformation matrix M in a *translation component* M_ρ for the motion along hole axis the hole $\bar{\boldsymbol{x}}_h = R_h \bar{\boldsymbol{u}}_x$ (wrt the base-frame), re-defined as $\bar{\boldsymbol{u}}_r$, and a *rotation component* M_θ for the angular pose correction about the axes of the plane orthogonal to the hole, $\bar{\boldsymbol{y}}_h$ and $\bar{\boldsymbol{z}}_h$ (wrt the base-frame)

$$M(t) = M_\rho(t) M_\theta(t).$$

The translation component

$$M_\rho(t) = \begin{bmatrix} I_3 & \rho_r(t)\bar{\boldsymbol{u}}_r(t) \\ \mathbf{0}^T & 1 \end{bmatrix}$$

has a translation step $\rho_r(t)$ that is the superimposition of a *damped-forward* translation $\rho_{r_+}(t)$, a *spring-backward* translation $\rho_{r_-}(t)$ and a *contact-force compensation* term ρ^*

$$\rho_r(t) = \rho_{r_+}(t) - \rho_{r_-}(t) - \rho^*.$$

The first one is a translation toward the hole axis with a variable step $\rho_{r_+}(t) \in (0, \Delta_a] \subset \mathbb{R}^+$ inversely proportional to the angular errors e_{θ_y} and e_{θ_z} through a gain k_f, and scaled by the distance error e_d saturated in $[0, 1]$

$$\rho_{r_+}(t) = \frac{\Delta_a}{1 + k_f \left[e_{\theta_y}^2(t) + e_{\theta_z}^2(t) \right]} \text{sat}(\beta e_d(t), 0, 1), \tag{4.2}$$

where $\text{sat}(x, x_{min}, x_{max})$ is a saturation function limited in $[x_{min}, x_{max}]$, while $\beta > 1$ is a scaling factor. Let us consider β big enough to keep $\text{sat}(\beta e_d, 0, 1) = 1$ for almost all the task and bring it to zero only when $e_d \simeq 0$, in order to turn off the forward translation action when the tool is close to the desired distance.

The second translation is on the opposite direction of the hole axis regulated by a PI action on the force

$$\rho_{r_-}^u(t) = k_{r_P} e_{\rho_r}(t) + k_{r_I} \int_0^{\Delta_t} e_{\rho_r}(t - \tau) d\tau, \tag{4.3}$$

that is then saturated between $(0, 2\Delta_a]$,

$$\rho_{r_-}(t) = 2\Delta_a \text{sat}\left(\frac{\rho_{r_-}^u(t)}{2\Delta_a}, 0, 1 \right) \in (0, 2\Delta_a] \subset \mathbb{R}^+. \tag{4.4}$$

Finally the contact-force compensation is a feed-forward term, $\rho^*(t) = (k_{r_P} + k_{r_I}\Delta_t) f_r^*$, depending on the desired contact force f_r^*.

4.5.2 Rotation Control

The rotation component is a concatenation of two rotations by the angles of $\theta_y(t)$ and $\theta_z(t)$ about the hole axes $\bar{\boldsymbol{y}}_h = R_h \bar{\boldsymbol{u}}_y$ and $\bar{\boldsymbol{z}}_h = R_h \bar{\boldsymbol{u}}_z$ respectively

$$M_\theta(t) = Q_h \tilde{R}_{\bar{\boldsymbol{y}}_h}(\theta_y(t)) \tilde{R}_{\bar{\boldsymbol{z}}_h}(\theta_z(t)) Q_h^{-1} \tag{4.5}$$

where $\tilde{R}_{\bar{\boldsymbol{i}}}(\theta)$ is the homogeneous transformation employing the pure rotation of an angle θ about the axis $\bar{\boldsymbol{i}}$. Since the transformation Equation 4.5 is done on the base-frame we first have to change the coordinates to the hole-frame by Q_h^{-1}, then apply the rotations, and finally change back to base-frame by Q_h.

The rotation angles $\theta_y(t)$ and $\theta_z(t)$ are PI actions on the estimated force feedback

$$\theta_i(t) = k_{i_P} e_{\theta_i}(t) + k_{i_I} \int_0^{\Delta_t} e_{\theta_y}(t-\tau) d\tau \tag{4.6}$$

where $i \in \{z, y\}$. Observe that in the angular correction we use as feedback the y and z components of the force on the tool-frame, through the errors $e_{\theta_y}(t)$ and $e_{\theta_z}(t)$, while for the linear correction we take, through the error $e_{\rho_r}(t)$, the absolute value of this force projected onto the hole longitudinal axis $\bar{\boldsymbol{x}}_h$ (thus we use the x component on the hole-frame).

4.6 Experimental Results

The proposed control strategy was validated through a series of insertions of electric wires ($3.5mm$) in a component simulacrum hole ($5mm$), visible in Figure 4.4. In these tests the wire does not reach the bottom of the hole, thus we set a null equilibrium contact force $f_r^* = 0$, while the desired distance takes into account the fingers size and is set at 1.5 cm, i.e. $\Delta_p^* = 0.015\,\text{m}$.

In order to implement the controller presented in section 4.5 in the experimental setup, we make a discrete-time approximation. Hence, a generic time-dependent function $g(t), t \in \mathbb{R}$, is replaced by its sampled version $\underline{g}(k) = g(Tk), k \in \mathbb{N}$, where $T \in \mathbb{R}^+$ is the sampling period, set to $T = 10ms$. The integrals in the control actions Equation 4.3 and Equation 4.6 are approximated as $\int_0^{\Delta_t} g\left(kT - \tau\right) d\tau \approx \sum_{i=0}^{\lceil \Delta_t / T \rceil} g(kT - i)T$.

Since the equilibrium would be reached asymptotically, in the experiments we consider the task completed when the errors are around the origin $e_{\rho_r} \in B_{\varepsilon_{\rho_r}}(0), e_{\theta_y} \in B_{\varepsilon_{\theta_y}}(0), e_{\theta_z} \in B_{\varepsilon_{\theta_z}}(0), e_d \in B_{\varepsilon_d}(0)$ with radius respectively given by $\varepsilon_{\rho_r}, \varepsilon_{\theta_y}, \varepsilon_{\theta_z}, \varepsilon_d \in \mathbb{R}$.

4.6.1 Experimental Setup

The hardware setup used during the experiment here described is shown in Figure 4.4. The system is composed by an industrial manipulator, a COMAU Smart Six, equipped with a commercial gripper, a Schunk PG70 electric parallel gripper. A custom tactile sensor developed in [98] has been mounted on one gripper's finger to provide a tactile image of the grasped objects. It is constituted by 16 taxels organized as a 4×4 matrix and a deformable layer with a flat shape. Each taxel is constituted by a single SMT photo-reflector integrating both an infrared LED and a PhotoTransistor (PT). An Arduino-based µcontroller board is then used to send the data to the control PC via USB connection. The control system has been developed

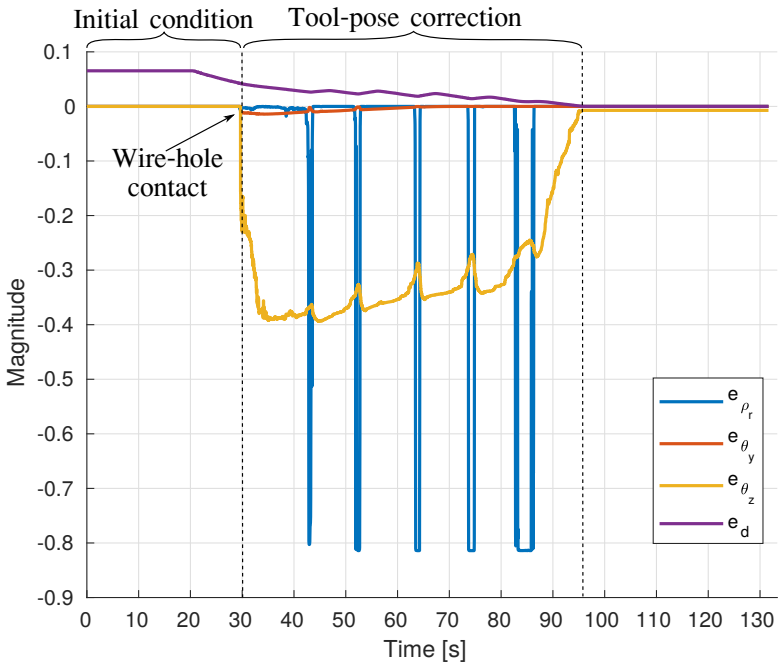


FIGURE 4.5: Errors signals during the insertion task.

exploiting the ROS middleware to allow the communication between the different parts (sensors, robot, gripper, etc.).

4.6.2 Results

The sequence of insertion is shown in Figure 4.4 for the configuration C2. The tests are run 12 times for the configuration C1, Figure 4.1a, and 17 times for the configuration C2, Figure 4.1b, with a success rate of 83.3% and 76.5% respectively. These two configuration are identified as the most likely deformation in a elastoplastic wire considering a realistic scenario. In each test the wire was re-grasped in a different position and with different lengths, while the wire was bent manually with a different deformation (in the two scenarios of Figure 4.1). The few failures reported during the experiments can be attributed to the following two main reasons: 1) the deformation curves were too pronounced, i.e. close to 90 deg; 2) the wire was grasped too far from the tip. In fact, when the curve is close or over 90 deg, the forces acting on the wire and estimated on the grasping point are in directions opposite to the needed correction. The backward translation action Equation 4.4 has been added to the control to prevent the wire to buckle between hole and the gripper. Unfortunately this situation might still happen when the deformation angle is close to 90 deg and the friction between the wire and the hole is sufficiently large (in the considered setup the wire is covered by a rubbery coating which produce an very large friction). To correct such issue might be sufficient increase the backward translation gains k_{r_p} and k_{r_I}. Another limit that rises always in DLOs manipulation based on force feedback is the difficulties to detect the external forces acting on the object when they are far from the measurement point. Hence, when the wire is grasped too far from tip the estimation of the contact forces become inefficient.

In Figure 4.5 the error signals from the insertion task shown in the sequence of Figure 4.4 are reported. Since the experiment starts with a simple approach toward the hole without any contact (initial condition), the error position e_d linearly

decrease. When the first contact takes place (wire-hole contact), the magnitude of the force error e_{θ_z} increases in negative value, due to the deformation C2 of the wire under consideration. As consequence, the tool starts rotating around $\bar{\boldsymbol{y}}_{l_1}$ (tool-pose correction), while it keeps moving toward the hole. This forward movement ensures to maintain the contact between wire and hole, in this way the tool continue rotation until the wire is not aligned with the hole. In fact, the error decreases significantly only in the last phase when the wire is aligned with the hole and the remaining part of the wire can be inserted straight. The error e_{ρ_r} increases when the wire is buckling, resulting in a backward translation on the tool, while the rotation continue. In fact, notice that e_{θ_z} decrease, if the task preceded without the backward translation, this error would become positive and the tool would change the rotation on the opposite and wrong direction ending in a failure. Hence, a trad-off between the forward and backward translation must be found adjusting the gains k_{r_P}, k_{r_I} and k_f, in order to keep contact between wire and hole but without pushing too hard (and buckling the wire).

4.7 Conclusion

In this work an automated robotic system for the insertion of a DLO inside a hole has been developed. We have provided the robot with a controller capable of correcting the orientation of the target cable during insertion to avoid bending; the controller is fed with feedback coming from a tactile sensor processed by an RNN able of regressing the 3 components of the force acting on the cable, including the force tangential to the sensor walls. The experimental results have shown that this control scheme is robust at various initial cable deformations (single and double bending). Future activities will be devoted to the integration with a model of the DLO in order to extend the manipulation capability to more complex wire deformations and obtain an higher success rate. Moreover, a formal proof of stability for the controlled system will be provided.

Chapter 5

A Robotic System for DLOs Reshaping in Cluttered Backgrounds

In the previous chapters, which constitute the first part of this thesis, we investigated the robotic manipulation and sensing of DLOs for grasping and inserting the terminals in a hole. In this and the next chapters instead, the attention is moved on DLOs reshaping. Hence, similarly to the first part, we begin presenting a full system for manipulate a DLO, while in the the next chapters we propose further improvements and extensions to the DLO sensing.

The system presented in this chapter performs the manipulation of the DLO with sequence of pick-and-drop primitives driven by visual data. We start by studying the manipulation of an highly deformable object, i.e. a soft rope, with a single robotic arm. In this case we can neglect the elastic component of the DLO e assume it is purely plastic. The proposed solution relies on a decision-making process which learns the optimal grasping location exploiting deep Q-learning and finds the best releasing point from a path representation of the DLO shape. Hence, also this application the DLO sensing represents a fundamental component for the whole system, in fact we employ a state-of-the-art algorithm for building a geometrical model of the DLO.

5.1 Introduction

Earlier works on deformable object manipulation have sought open-loop strategies, which are ineffective since the material can shift in unpredictable ways [103]. Successive works attempted to develop various model-based strategies for controlling the object shape through robot manipulation [90, 104]. This is a common and effective approach with rigid objects, but it results weak with non-rigid objects. Indeed, there is no obvious mapping from an observation of the object to a compact representation in which planning can be performed.

Deep Reinforcement Learning (DRL) is becoming more and more popular in robotic manipulation [105, 11, 19, 106, 107, 108]. We are actually witnessing a run for the best DRL algorithm (in terms of flexibility and efficiency), that would enable the robot to perform any kind of manipulation, without engineering but only through its personal interacting experience with the environment [109]. However even the state-of-the-art solutions based on DRL algorithms produce results [106, 107, 108] quite far from those achievable with classical engineering methods.

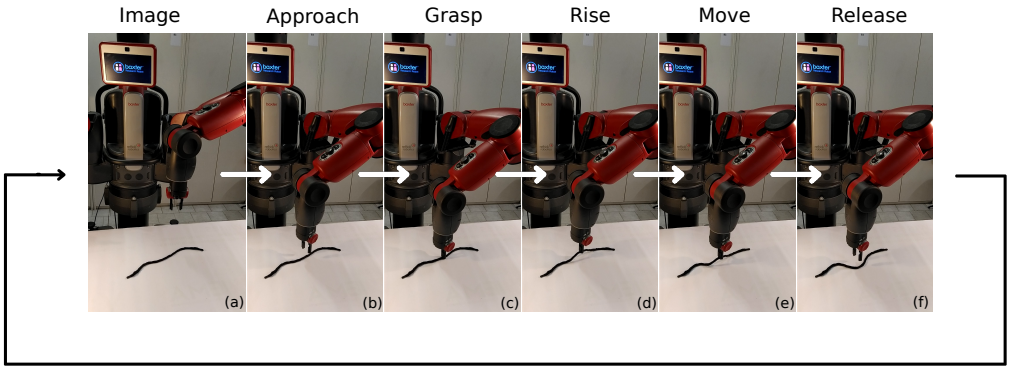


FIGURE 5.1: Pick-and-drop trajectories performed by the robot during every iteration of the proposed algorithm. It starts acquiring an image of the table with the hand camera (a). The decision process select a grasping point based on this image, and compute the corresponding releasing point. In the second step the right arm move toward the grasping point and stops $0.05m$ over the table (b). Then it grasps the DLO in the decided point (c) and returns to the approaching point (d). In the last two steps approaches the releasing point (e) and opens the gripper (f). Finally returns in the initial configuration ready to start over, by taking a new image (a).

The challenge in these works is the development of an algorithm which could learn the joint torque trajectories for a generic task directly from the input raw images by means of a rewarding system. This process demands to the agent to intrinsically learn operations like inverse kinematics, trajectory planning, visual feature extraction, object detection and semantic segmentation. All problems extensively studied and efficiently solved in literature.

Anyway, in order to discard the requisite of a model, one of the major challenge when interacting with deformable objects, reinforcement learning seems a very reasonable and very attractive approach [19, 110]. In fact, the optimization skills and the flexibility of DRL are essential to overcome the complex behaviour of deformable objects. However, to the state-of-the-art of DRL, a worth solution would be lighting the learning load by integrating the DRL algorithms with other non-learning-based tools and engineering consideration, in order to make the most of their capabilities.

In this work, we build a smart integration between efficient engineered solutions and DRL algorithms. In particular we propose a wise use of DRL algorithms in the few tasks in which the process needs to predict the optimal interaction with the DLO. While we prefer to employ a stable inverse kinematics (IK) solver and a trajectory planner to perform the robot motion. Moreover, we lighten the information extraction from visual data with a state-of-the-art vision technique specifically designed for DLOs [10]. The presented work is motivated by the lack of effective application solutions for DLO manipulation in tasks like untangle, spread and routing a wire in assembly processes [37, 111]. Thus, our study wants to move a step forward into these challenging tasks, proposing a solution able to control the shape of a DLO in a clutter environment using vision feedback.

In line with our work, also other authors adopted similar approaches. Boularias et al. [112] explores the use of DRL combined with well-known techniques for image segmentation, for manipulating unknown objects. They propose a pipeline that first segments images into separated objects, predicts pushing and grasping actions,

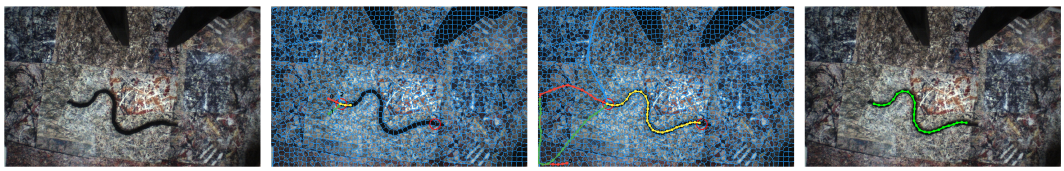


FIGURE 5.2: Image segmentation algorithm for DLOs. The first step consists in segmenting the input image into adjacent sub-regions (superpixels) and creating an adjacency graph. From the extremity of the DLO, an arbitrary number of walks are started, by moving into adjacent superpixels. Each walk moves forward along the adjacency graph by choosing the best next superpixels until it reaches the other extremity (this walk is masked as 'closed'). As a set of random walks are started, Ariadne keeps only the most likely one, among those marked as 'close'.

extracts hand-tuned features for each action, then executes the action with highest expected reward. In [20] and [113], to make training tractable on a real robot, they simplified the action space to a set of end-effector-driven motion primitives. They formulate the task as a pixel-wise labeling problem: where each image pixel – and image orientation – corresponds to a specific robot motion primitive executed on the 3D location of that pixel in the scene. Similarly to these works, we turn action prediction into a classification problem by discretizing the action space and we define specific robot motion primitive (grasping and releasing).

The main contributions of our work are: (1) a novel robot learning-based system for autonomous deformation of a rope from/to a general shape using visual feedback capable to work with any cluttered background; (2) a study on DLOs deformation through a re-positioning sequence, in particular we investigated different strategies to decide the grasp/release locations and their relations.

The remainder of this chapter is structured as follows: section 5.2, reports an overview of previous works in this field; section 5.3 presents the experimental setup; section 5.4 provides relevant background on reinforcement learning and Deep Q-Network; section 5.5 describes the proposed method in details; finally, in section 5.6, we examine the experiments and make some piratical considerations.

5.2 Related Works

The problem of DLOs manipulation has been studied before, with particular attention to tying knots. For instance, Yamakawa et al. [114] proposed a trajectory planning approach where a knot can be tied with a single robot arm at high speed. Mayer et al. [115] examined the use of recurrent neural networks to learn the knot tying trajectories. Learning from Demonstration (LfD) was proposed by Lee et al. [116] to learn a function that maps a pairs of correspondence points, while minimizing a bending cost.

The insertion of a DLO in a hole is another widely investigated task, due to all the useful applications that it would have in assembly operations [7, 111]. Inaba et al. [88] developed an hand-eye system to insert a rope into a hole using stereo vision for computing the relative position between rope tip and hole. In [117] they presented a method to insert string through tight workspace openings online using an approximate Jacobian to estimate the motion of the string. In [8] the insertion of a DLO into a hole is performed by analyzing the feedback coming from a tactile

sensor by means of a recurrent neural network which estimate the force acting on the wire itself.

Few works attempt to address the shape control of a DLO using a robot. Rambow et al. [118] used a two-arm robot to mount a deformable tube in a desired configuration based on a single teleoperated demonstration. Nair at al. [20] developed a learning-based system where a robot takes as input a sequence of images showing small deformations of a rope from an initial to the goal configuration, performed by a human demonstrator, and outputs a sequence of actions that would lead the rope to the target shape, imitating the demonstrator deformations sequence. In [20] a Baxter robot has been configured to collect interaction data with the rope for 500 hours, used later to learn an inverse dynamics model which is finally employed to imitate the human demonstration. Similarly, also Sundaresan et al. [21] proposed an approach using imitation learning to arrange the configuration of a rope. They also show that the proposed solution can be used for a knotting task from human demonstration and assuming to start always from the same configuration containing a single loop. To brake symmetry and enable consistent correspondence mapping with target shape in [21] and [119] added, respectively, a ball and a blue tape. Moreover, in [119] they also tied one end of the rope to a clamp attached to the table. In this work, instead, we use a perfectly symmetric rope, with both the extremity free and identical. Another recent work on the same topic is [16], where they estimate a state-space representation of the rope and learn a dynamics model with an LSTM network and solve the rope manipulation with MPC. The weakest point of this solution is the assumption of having a strong color contrast between the rope and the table for a correct state estimation.

Differently from the over mentioned works, we addresses the problem of autonomous deformation of a rope from/to a general shape by training a reinforcement learning agent from scratch on a real robot, without: (1) the necessity of demonstrate the intermediate deformation steps in test time; (2) adding easily distinguishable object to brake the rope symmetry; (3) fixing any extremity to the table; (4) making any restrictions on the background color. In the sequence of Figure 5.4 we used a white background to make images clearer and to facilitate readers in the vision of the rope. However, as explained in subsection 5.5.2, the system is designed to work on heterogeneous and confusing backgrounds.

5.3 Experimental Setup

For the experiments described in the work, we employ a Rethink's Baxter robot, which has a wrist-mounted gripper with two degrees of freedom (one rotational and one for closing/opening the two fingers). An RGB camera integrated with the robot hand provides visual data, with a resolution of 960×600 px.

The setup is illustrated in Figure 5.1. Also in this case, a white background is used to make images clearer and to facilitate readers in the vision of the rope. However, it is worth to remark that, as explained in subsection 5.5.2, the system is designed to work on heterogeneous and confusing backgrounds, see e.g Figure 5.2.

A perfectly symmetric DLO (i.e. a rope), lies free on a table, at a known height z_*, in front of the robot. We define a fixed camera pose over the table to acquire the input RGB image. The interaction of the robot with the rope is limited to two simple motion primitives consisting of grasping the rope at location (u_1, v_1) and releasing it at location (u_2, v_2), where u_1, v_1, u_2, v_2 are pixel coordinates in the input RGB image. Since both the table height and hand-camera pose are known with respect to the

robot base frame, we can estimate the grasping (x_1, y_1, z_*) and releasing (x_2, y_2, z_*) coordinates in the base frame.

As shown in Figure 5.1, during the grasping the robot first approaches the point (x_1, y_1, z_*) from the top, with and offset of $z' = 0.05\,\text{m}$ along the vertical z-axis and the gripper open. It moves down with a linear trajectory in the Cartesian space along z to z_*, then it close the gripper's fingers before rising back to $z_* + z'$. The motion sequence for dropping the rope is the same, with the intuitive difference that it starts with the gripper close, and opens it after the descent to z_*. In both the motion primitives, the motion planning is automatically executed with the native Baxter's IK solver.

5.4 Preliminaries on DRL

We formulate the grasping task as a Markov decision process defined by $(\mathcal{S}, \mathcal{A}, p, r)$. Where state space $\mathcal{S}$ and action space $\mathcal{A}$, that represent respectively all possible combination of current and target shape and all possible grasping point in the scene, are assumed to be discrete. In subsection 5.5.2 and subsection 5.5.4 we illustrate the discretization strategy and we define the environment's state, while in subsection 5.5.5 we define the agent's actions. The unknown state transition probability $p(s_{t+1}|s_t, a_t)$ represents the probability density of the next state s_{t+1} given the current state s_t and current action a_t. For each state s_t at time t of the environment (i.e. the DLO), the agent (i.e. the robot) chooses and executes an action a_t according to the policy $\pi(a_t|s_t)$, which implies the transition of the environment to a new state s_{t+1} and the formulation of a reward r_t as defined in subsection 5.5.4. Under this formulation, the goal is to find an optimal policy π^* that maximizes the expected sum of future rewards $\sum_{t=i}^{+\infty} \mathbb{E}_{(s_t,a_t)\sim p_\pi}[r_t]$, where we use ρ_π to denote the state or state-action marginals of the trajectory distribution induced by a policy $\pi(a_t|s_t)$.

In this work, we investigate the use of deep Q-learning, that is a Q-learning where a deep neural network is used to approximate the Q-value function $Q_\pi(s_t, a_t) = \sum_{t_i=t}^{T} \mathbb{E}_{\pi_\theta}[r_t|s_t, a_t]$, which measures the expected reward of taking action a_t in state s_t at time t. The network that approximates Q-value function is called Deep Q-Network (DQN) [120] and the training data are processed by using stochastic gradient updates. In Q-learning, a greedy policy $\pi(a_t|s_t)$ is trained to choose optimal actions by maximizing the action-value function $Q_\pi(s_t, a_t)$. Formally our learning objective is to iteratively minimize the temporal difference error δ_t of $Q_\pi(s_t, a_t)$ to a fixed target value y_t,

$$\delta_t = |Q_\pi(s_t, a_t) - y_t|$$
$$y_t = r_t + \gamma\, Q_\pi\left(s_{t+1}, \underset{a' \in \mathcal{A}}{\text{argmax}}\, Q_\pi\left(s_{t+1}, a'\right)\right)$$

where $\gamma \in \mathbb{R}_+$ is called the discount rate.

5.5 Method

5.5.1 Overview

In this section, we describe our method to reshape a DLO using a single arm robot. The proposed method relies on a DQN-based decision process that leverages on an effective visual representation of the DLO shape. Current and target shapes are

modeled using both a *Key Points Path* and a *Spatial Grid Matrix*, detailed in subsection 5.5.2. The interaction with the DLO, and its reshaping process, take place through a sequence of grasping and releasing operations. The decision process, detailed in subsection 5.5.3, learns to predict the best grasping point from the input image while the corresponding releasing point is computed by projection. A sample sequence of steps that leads the DLO to the target shape is shown in Figure 5.4. Since the proposed method relies on a reinforcement learning algorithm, in subsection 5.5.4 and subsection 5.5.5, we formally define states, actions and rewards, while in subsection 5.5.6 some considerations about the training and how we speed it up when starting from scratch are made.

5.5.2 Shape Representation

In order to effectively exploit its decision-making skills, the DRL agent has been integrated into a framework that lightens the learning load, as will be detailed in subsection 5.5.3. This process is based on two representations of the DLO, both shown in Figure 5.3, processed from the visual input. The first representation, consists of a sorted sequence of key points belonging to the DLO. This representation allows us to effectively identify the releasing point on the target shape as a projection of the grasping point (taken from the current shape). In this way the agent needs to learn only the grasping point. In the second representation, a dimensionality reduction of the visual data is performed by mapping the segmentation mask into a spatial grid matrix. This matrix will later compose the state of the environment that the agent uses to predict the best action to perform.

Both the representations relays on a algorithm called Ariadne [10], able to perform simultaneously instance segmentation and b-spline modeling of DLOs. The basic idea of Ariadne is to detect the DLOs as suitable walks over the Region Adjacency Graph built on a super-pixel over-segmentation of the source image. In Figure 5.2 is visible an example of segmentation on a cluttered background.

Key Points Path

Ariadne segments the image into adjacent sub-regions (superpixels) then finds a walk that connects the two extremities of the DLO. This walk is essentially a sorted list of superpixels, that can be represented by their centroids, hence it can be converted into a sorted list of image points $\mathcal{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_n]$. Each walk need to be initialized with seed superpixels located at the DLOs' extremities. Purposely, we deployed YOLO v2 [121], an object detection tool based on convolutional neural networks. We fine-tuned the YOLO v2 model, pretrained on ImageNet, on a dataset that we created with the black rope used in the experiments. To create this dataset we developed an automated labeling tool based on video sequences that we allows us to easily gather massive amounts of training images in the field with minimal human intervention [60]. The tool is based on the idea that restricted camera movements (i.e. lift and rotate) leads to a controlled rigid transformation $\mathbf{A}$ between the two consecutive images I_i, I_{i+1} such that $I_{i+1} = \mathbf{A}I_i$. The same rigid transformation A can be applied to each bounding box (BB) $\check{b}_i$ present in the image I_i so as to obtain a new set of BB such that $\check{b}_{i+1} = \mathbf{A}\check{b}_i$. This procedure can be repeated for each consecutive pair of images in the video sequence, it is therefore clear how the sole human intervention is to create the BB labels in the first frame I_0.

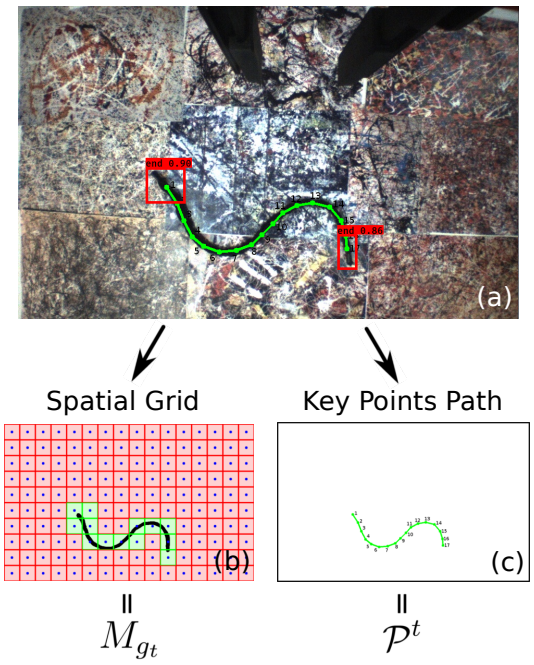


FIGURE 5.3: The input raw image is processed by Ariadne (a). Since it needs to be initialized with the DLO extremities, YOLO object detector is employed for the purpose. Ariadne produces a binary mask and a list of image points that describes a walk along the DLO. From the binary mask we create the spatial grid (b) and define the matrix M_{g_t}, while from the points path (c) we obtain the list of points $\mathcal{P}^t$

Spatial Grid Model

A uniform space partitioning is performed on a binary image mask $I_t^{\text{mask}} \subseteq [0,1]^{h \times w}$ obtained as segmentation of the DLO from the input RGB image $I_t \subseteq [0,255]^{3 \times h \times w}$. This partitioning consists of a set with size $n_{\text{rows}} \times n_{\text{cols}}$ of rectangular regions of pixels $\{\Psi_{i,j} \in \mathbb{R}^{\psi_h \times \psi_w}\}_{i \in n_{\text{rows}}, j \in n_{\text{cols}}}$ (image windows) with constant size $\psi_h \times \psi_w = \frac{h}{n_{\text{rows}}} \times \frac{w}{n_{\text{cols}}}$. Each region is mapped into a scalar value $g_t^{i,j} = \Omega_{[0,1]}\left(\frac{1}{\psi_h \psi_w} \sum_{u,v \in \Psi_{i,j}} I_t^{\text{mask}}[u,v], g^{\text{Th}}\right)$, that is the average of all the region-pixels binarized through the function $\Omega_{[0,1]}(x, x^{\text{Th}})$, which gets 1 only when $x \geq x^{\text{Th}}$ and 0 otherwise. From these values we define the spatial grid matrix at time t as $M_{g_t} = [g_t^{i,j}]_{i \in n_{\text{rows}}, j \in n_{\text{cols}}} \in [0,1]^{n_{\text{rows}} \times n_{\text{cols}}}$, where every cell (i,j) and every region $\Psi_{i,j}$ have a bijective correspondence. To simplify position calculations, each region is represented by its center point.

5.5.3 Decision Process

The goal is to reshape a DLO by means of a sequence of grasp and release operations. To achieve this we employ the decision-making process schematically outlined in Figure 5.5. This process aims to determine the optimal grasping and releasing points, respectively $p_{\text{grasp}} \in \mathbb{R}^2$ and $p_{\text{release}} \in \mathbb{R}^2$, in order to maximize the visual overlap between the current and the target shapes, using as input data the image of the current scene.

A straightforward approach that we initially explored is to train an agent for learning jointly the two optimal locations p_{grasp} and p_{release} from the observation of the current scene s_t. However, the releasing location is strongly dependent on the

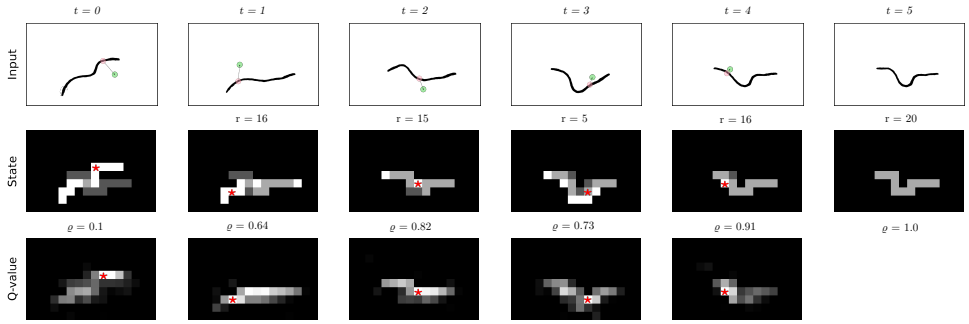


FIGURE 5.4: Example of grasping-and-releasing sequence for reshaping the DLO. From the target and the current input visual data (first row) we define the state s_t (second row) and predict the Q-value $\phi_Q(s_t)$ (third row). We use a visual representation of the state where a cell can be: black if part of the background ($s_t^{i,j} = 0$); white if part of the current shape only ($s_t^{i,j} = 2$); light-grey if part of the overlapped region ($s_t^{i,j} = 3$); dark-grey if part of the target shape only ($s_t^{i,j} = 1$). In each step t, we obtain the action a_t as the coordinates to the highest value of $\phi_Q(s_t)$ (red star). On the input images we draw the grasping (red circle) and the releasing (green circle) points correspondent to the predicted action a_t. For each transaction we also compute the reward $r(a_t, s_t, s_{t+1})$, as a function of the overlap score $\varrho(s_t)$ (see Equation 5.3).

grasping point, but the over mentioned approach does not take into account this conditional nature of the two operations.

To address this problem, we could combine two agents in a cascade, where the first predicts the grasping point and the second the releasing point. In other words, instead of learning jointly the two locations with an unique policy $\pi([p_{\text{grasp}}, p_{\text{release}}]|s_t)$, we define two policies: one that learns the grasping point from the current state $\pi_{\text{grasp}}(p_{\text{grasp}}|s_t)$; while the other one learns the releasing point from both the state and the predicted grasping location $\pi_{\text{release}}(p_{\text{release}}|s_t, p_{\text{grasp}})$. Nevertheless, training this policy is inefficient. In fact, the two operations would require two dedicated rewards, but we can only generate one reward after the releasing which is proportional to the visual overlap between the current and the target shapes. Clearly, in this setup, the decision process does not have the possibility to understand if an high (or low) reward is due to π_{grasp} or π_{release}.

Ultimately, to overcome also this action reward assignment issue, we propose to only learn the grasping point, while the releasing point is derived from the *key points path* representation of current and target shapes presented in subsection 5.5.2. In fact, given the target shape path $\mathcal{P}^* = [\mathbf{p}_1^*, \dots, \mathbf{p}_m^*]$ and the current shape path $\mathcal{P}^t = [\mathbf{p}_1^t, \dots, \mathbf{p}_n^t]$, we can easily project a point form one path to the other. In particular we can project the grasping point $\mathbf{p}_k^t$, taken from $\mathcal{P}^t$, into a releasing point $\mathbf{p}_s^*$ belonging to $\mathcal{P}^*$, where $s = \left\lfloor k\,\frac{m}{n} + \frac{1}{2} \right\rfloor$.

Having established that we can find the placing location with this projection strategy, one might wonder if we can choose also the picking point simply form the representations, without learning it. The most trivial solution would be grasping every time a random point from those that are not overlapped with the target shape. This is clearly very ineffective, since it neglects the DLO property of interconnection among the key points. Moreover, if we grasp only the free points, i.e. those that are not-overlapped, we can not ensure to really reshape the rope, since

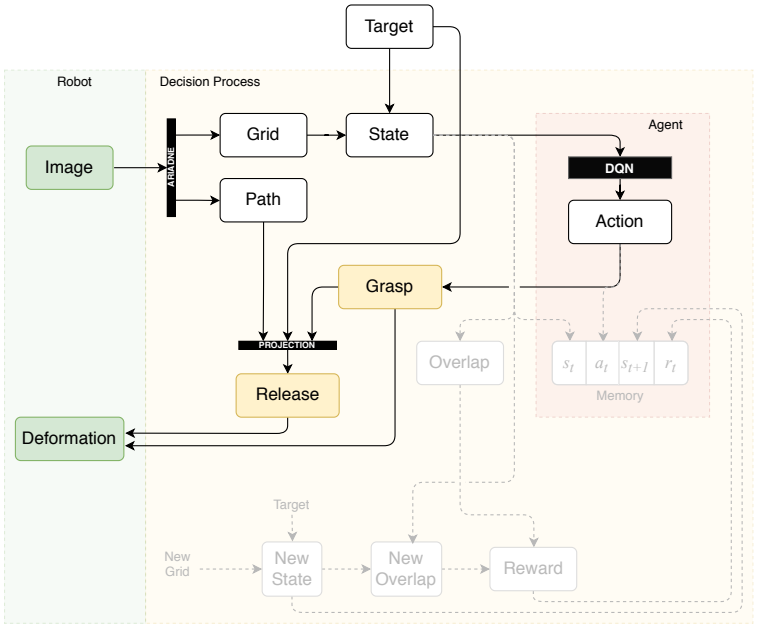


FIGURE 5.5: Scheme representing the proposed method. We highlight in green the *robot side*, which includes the image acquired by the hand camera and the deformation (grasp and releasing operations) executed on the DLO. The decision making process is highlighted in yellow and the agent in red. The scheme shows also the agent's memory update, with dashed lines and grey boxes. In particular, the bottom part of the scheme reports the *new state* and *new overlap* that are obtained from the same scheme in the successive time step, from the new image acquired after the deformation.

the algorithm would simply aim to clean all the free points moving them to the target location. Hence, a trivial solution such as winding the rope in a small region that completely overlaps just a portion the target would conclude erroneously the task if there no free point is left. On the other hand, if we grasp also those already overlapped, we risk to make many pointless re-positioning actions. Another trivial approach would be following the order in the path, but also in this case we are not taking into account the interlinked nature of the object. In fact every time we place a point we might erroneously move those that we placed earlier.

As already stated, in the proposed solution we develop a decision process based on a DQN agent that learns the optimal grasping cell (action) in a grid that combine the spatial information of both the target and the current shapes (state). As shown in Figure 5.5 the agent is wrapped into a structure that defines the agent's state by extracting the useful features from the input image and derives the grasping and releasing point from the agent's action. The task starts by providing a goal that can be either a key points path and a spatial grid or a raw image of the rope in a target shape. In each iteration the system acquires a new RGB image of the scene. Then, the visual segmentation algorithm, creates the binary mask and the key points path for the current shape. The mask is reduced to the correspondent spatial grid matrix, which is combined with the target's one, as defined in subsection 5.5.4, to obtain the state. The agent predicts the best action for the current state, i.e. it provides the optimal grasping cell of the spatial grid, as detailed in subsection 5.5.5. This action needs to be mapped into a grasping point with respect to robot frame $\{B\}$, so first we find the point in the input image as center of the region of pixels corresponding

to the grasping cell, and then, with the knowledge of the camera pose, we transform it with respect to $\{B\}$. The releasing point, as explained previously in this section, is obtained from the key points path and the grasping point. While the angle is simply estimated with a line fit algorithm from the image window contained in a the corresponding cell. Note that this estimation is affected by an ambiguity of π between current and target shapes. This would imply an undesired twist when releasing the rope. To have a consistent angle between the two shapes we can use the sorting information of the key points in the path. By consistently defining the two extremities on target and current shapes, the ambiguity is automatically solved. Obviously, arises a new problem on defining the extremities, since the DLO is perfectly symmetric. Let A and B be the end points of the current shape and A^* and B^* those of the target one. Thus, we define A^* as the end point of the target closer to A, which is instead arbitrarily assigned, and B^* the other one.

Once the robot has performed the deformation as explained in section 5.3, a new iteration starts. In the successive iteration, the reward, that is function of the overlap score, and the new state are computed and sent to the agent, which records the transaction state, action, new state and reward for the learning. The task ends when the overlap score reaches a given threshold.

5.5.4 Environment

We model each state s_t as a linear combination of the spatial grid matrix of the scene at time t, M_{g_t}, and the one of the target shape, M_{g^*},

$$s_t = 2M_{g_t} + M_{g^*}. \tag{5.1}$$

In this way the state is a matrix $s_t \in [0,3]^{n_{\text{rows}} \times n_{\text{cols}}}$ where each element $s_t^{i,j}$ corresponds to the cell (i,j) of the spatial grid built on the scene. Note that it can be rewritten as

$$s_t^{i,j} = \begin{cases} 0 \text{ if } \Psi_{i,j} \text{ is part of the background} \\ 1 \text{ if } \Psi_{i,j} \text{ is part of the target shape only} \\ 2 \text{ if } \Psi_{i,j} \text{ is part of the current shape only} \\ 3 \text{ if } \Psi_{i,j} \text{ is an overlapped region} \end{cases}, \tag{5.2}$$

where the overlapped regions are set of image pixels belonging to both the target and the current shape.

5.5.5 Agent

This work uses an implementation of deep Q-learning, where the DQN $\phi_Q(s_t)$ that approximate the Q-function $Q_\pi(s_t, a_t)$ is a convolutional neural network (CNN) schematically represented in Figure 5.6. Since both state and action space are quite simple by construction, simple network architectures can be used as well. The default architecture consists of five convolutional layers interleaved with nonlinear activation functions (ReLU) [119] and spatial batch normalization [122]. As already said, the input and the output of the DQN have the same size, that is the size of the spatial grid, $n_{\text{rows}} \times n_{\text{cols}}$.

Actions The agent predicts a vector action $a_t = [i\ j]^\top$, where $i \in \mathbb{N}^{n_{\text{rows}}}$ and $j \in \mathbb{N}^{n_{\text{cols}}}$ are the coordinates of a target region in the spatial grid where to perform

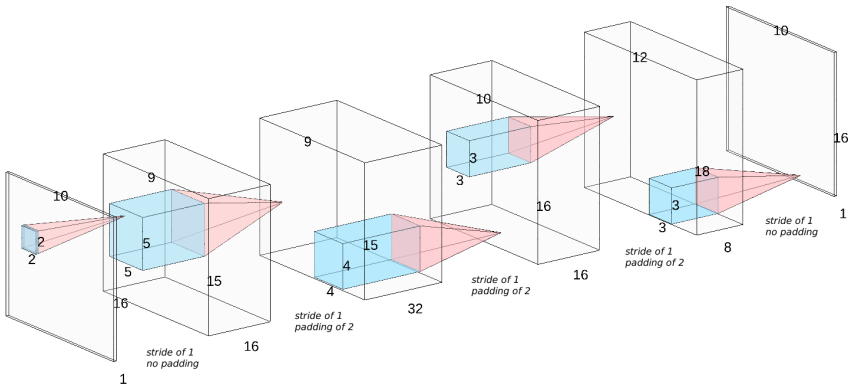


FIGURE 5.6: The DQN is a CNN with five convolutional layers, where input and output have the same size 10×16.

the grasping. These coordinates are easily inferred from the DQN's output $\phi_Q(\cdot) \in \mathbb{R}^{n_{\text{rows}} \times n_{\text{cols}}}$. In fact, the matrix ϕ_Q has the same size of the spatial grid matrix M_{g_t}, thus we have a one-to-one correspondence between the elements. This implies that we can take $\phi_Q^{i,j}(s_t)$, the value in coordinate i, j of $\phi_Q(\cdot)$, as the approximated Q-value $Q_\pi(s_t, a_t)$ of the action $a_t = [i\ j]^\top$, or in other words, $\phi_Q^{i,j}(s_t)$ can be considered as the expected future reward of grasping the DLO in the region (i, j). Hence, the action that maximizes the Q-function is the couple of indices corresponding to the region with the highest Q-value across the spatial grid matrix: $\text{argmax}_{a'} Q_\pi(s_t, a') = \text{argmax}_{(i,j)} \phi_Q^{i,j}(s_t)$.

Reward Shaping In our decision process we use a shaped reward. In fact, shaped reward functions compared to sparse reward functions, require more design effort as they incorporate knowledge of the problem into the reward structure, but in general they require less time to train, or at least they should speed-up the training in a complex setup.

The reward scheme we designed is very simple. First of all let us consider the state as written in Equation 5.2. We can easily assert that only the regions belonging to the current DLO shape are worth considering for grasping, which means that we can assign a reward $r(a_t, s_t, s_{t+1}) = 0$ to all the actions $a_t = [i\ j]^\top$ that leads the robot to the regions corresponding to the value $s_t^{i,j} \in \{0, 1\}$ or equivalently $g_t^{i,j} = 0$ (void grasping).

Let us consider now a valid action, $a_t \in \{[i\ j]\top : s_t^{i,j} \in \{2, 3\}\}$. Our goal is to maximize the number of overlapped regions, ideally the algorithm should converge to a state in which current and target shapes are completely overlapped in the spatial grid model, that is $s_t : s_t^{i,j} \in \{0, 3\}, \forall i, j$.

We define an overlap score $\varrho(s_t) = \frac{n_{s_t=3}}{n_{s_t \neq 0}}$ at time t as the number of overlapped regions $n_{s_t=3}$ over the number of all regions that are either part of the current or the target shape $n_{s_t \neq 0}$. Hence, assuming that $\varrho(s_{t+1}) - \varrho(s_t) > 0$, the reward that we assign to a valid action is directly proportional to the increment in the overlap score

$$r(a_t, s_t, s_{t+1}) = k \left[1 + \frac{\varrho(s_{t+1}) - \varrho(s_t)}{1 - \varrho(s_t)} \right], \tag{5.3}$$

where $k \in \mathbb{R}$ is a gain that we set to $k = 10$.

Moreover, to penalize the actions that cause an overlap loss, $\varrho(s_{t+1}) - \varrho(s_t) \leq 0$, we assign a constant reward $r(a_t, s_t, s_{t+1}) = \frac{k}{2}$, greater than zero (since the action is still valid) but always smaller than Equation 5.3.

5.5.6 Training and Test

We train the DQN using Adam optimization with fixed learning rates of 10^{-4}. Our models are implemented in PyTorch and trained with an NVIDIA GeForce GTX 1080 Ti on an Intel Core i7-7700K CPU clocked at 4.20GHz. We train with prioritized experience replay [123] using stochastic rank-based prioritization, approximated with a power-law distribution. Our exploration strategy is ϵ-greedy, with ϵ initialized at 0.7 then annealed over training to 0.1. Our future discount γ is constant at 0.5. The experience replay uses batches of size 132.

At the beginning of the training the DQN has random values and the agents can only take random actions in order to explore the environment. To speed this process up, human expertise can be used as agent's prior knowledge or heuristic. Hence, in the first phase of the training a human demonstrator provides a sequence of pick points on the rope toward the target shape, while the agent only collects data (i.e. state, action, reward and new state). Ideally, once the process is over, the agent has learnt a raw but satisfactory policy. Thus, in the second phase of the training, the agent can acts autonomously on the system and collects more self-generated data. Differently to other works like [20] or [21], the human demonstrations are used only to initialize the agent's experience and no longer needed in test time.

The demonstration phase is useful for gathering a large amount of meaningful data, possibly that cover a wide set of different scenarios. Hence, the demonstrator should to prevent the system to fall in some irrecoverable state (highly tangled DLO) or in a loop of similar transitions, that could cause an over-fit in the DQN training. For this reason, in the first phase, we change target shape as soon as we reach an overlap score of $\varrho^{\text{Th}} = 0.5$, since over this threshold we are performing small adjustments and the state would remain similar in a long sequence of transitions. This fine learning can be done in the second phase of autonomous exploration, when the agent has already some raw experience on the task. Following this principle we gradually increase the overlap score threshold ϱ^{Th} up to 0.8 every 50 transitions with step $\Delta\varrho = 0.1$.

We observed that, the agent first learns to find the non-empty regions taking into account that all the regions are linked because part of the same DLO and some of them are already correctly aligned with the target. In order to avoid over-fitting the agent on a particular shape, we collected 30 target shapes and change among them every $n = 15$ transitions or every time the overlap score reaches the given threshold.

5.6 Evaluation

In this section we evaluate the proposed method on our experimental setup. The spatial grid considered for the DLO shape representation has size $n_{\text{cols}} \times n_{\text{rows}} = 16 \times 10$. We collected 200 transactions by demonstration and other 300 during the autonomous exploration phase. We evaluate the performance by counting the number of steps required to reach an overlap score greater than 90% ($\varrho(s_t) > 0.9$). By running the experiment on 30 different scenarios, we estimate a success rate of 76.7% (23/30 tests) in achieving the goal with less than 12 steps and 86.7% (26/30 tests) with less than 18 steps. In 4/30 tests we assumed a failure due to an undesired

tangling. In Figure 5.7 and Figure 5.8 the 10 experiments are reported, showing the intermediate deformation steps performed by the robot and the agent's state. In this figure, the images have been binarized to improve readability. It is worth noticing that the system learns to stretch the DLO in only 2 steps by simply adjusting the two extremities.

The experimental data reported in Figure 5.4 show an example of correct learning, where the agent predicts as optimal grasping locations those that are not aligned with the reference shape. In particular, this behaviour is clearly visible in the first two steps and in the last one. Note also that the estimate Q-values are zero in the cells that are empty or occupied by the target shape only (not suitable for grasping). Moreover, while the cells not aligned with the reference are frequently preferred to those already aligned, these are not excluded, as happens in the 4th step of Figure 5.4.

5.7 Conclusions

In this work we studied the robotic manipulation of a deformable linear object lying on a table, i.e. a rope, using visual data. The proposed method relays on a decision making process that learns the optimal grasping location from the input visual data, by means of a DQN agent, and finds the best releasing point from a path representation of the rope shape. Also other solutions are examined and discarded for inefficiency or inadequacy. Differently from other studies in that field, the proposed technique only needs very limited human intervention during the initial training phase, while the system is able to learn autonomously how to deal with generic scenarios thereafter.

Experimental results of reshaping tests are provided, showing the intermediate steps of deformation that lead the rope from its initial configuration to the target and we examined the output of the DQN in each step of a sample experiment. This results show that our system is capable to manipulate ropes into a variety of different shapes in few steps.

Since our technique only assumes a Q-learning algorithm with CNNs, we believe it can be easily improved by applying state-of-the art algorithms, e.g. HER [124] or including some awareness of the sequential deformation by integrating recurrent neural networks.

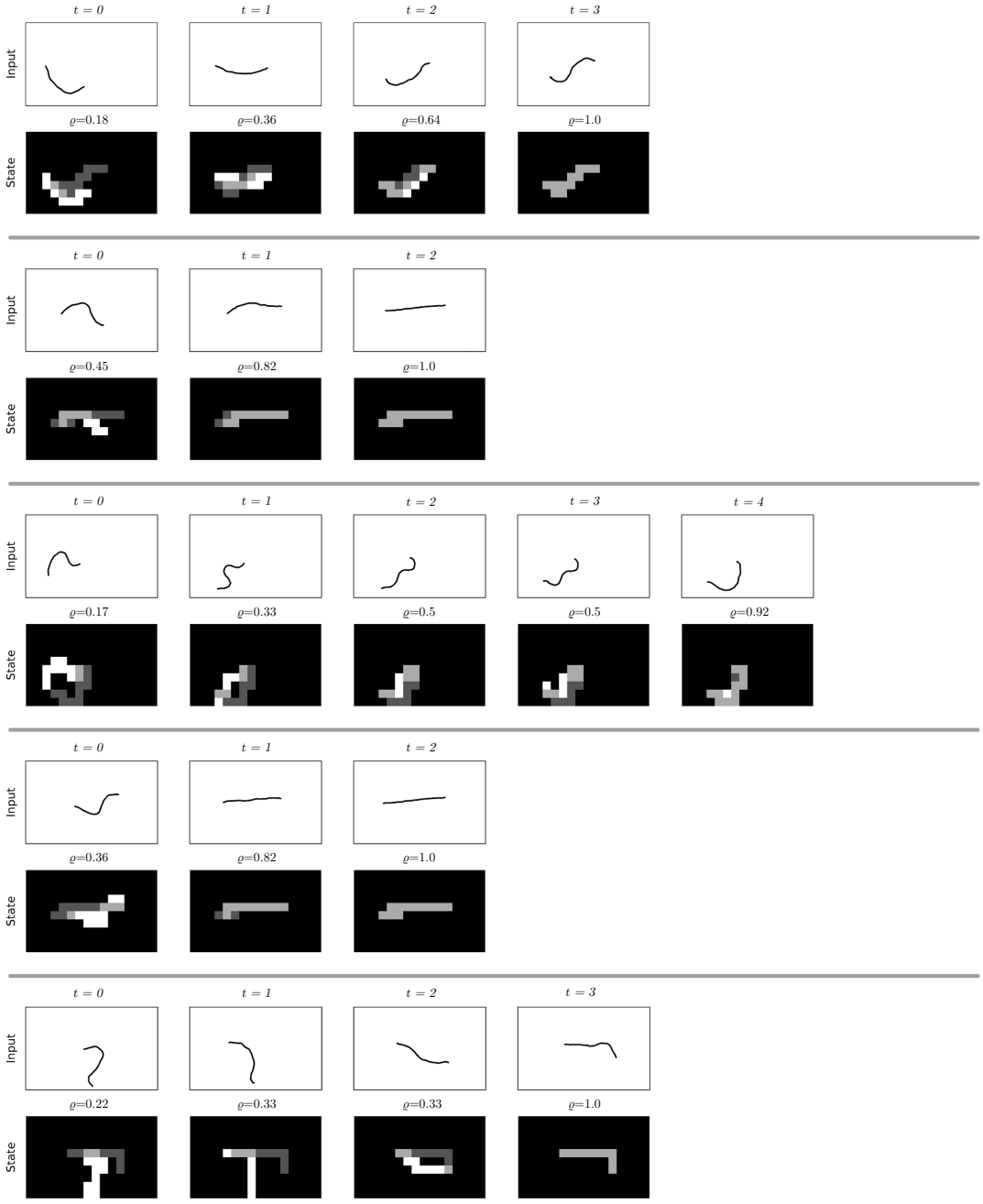


FIGURE 5.7: First set of 5 experiments that shows the DLO deformation steps performed by the robot using the proposed method. The images are binarized for visual clearance. The final shape corresponds to an overlap score greater than 90% ($\varrho(s_t) >= 0.9$). The state cells are: black if $s_t^{i,j} = 0$; white if $s_t^{i,j} = 2$; light-grey if $s_t^{i,j} = 3$; dark-grey if $s_t^{i,j} = 1$.

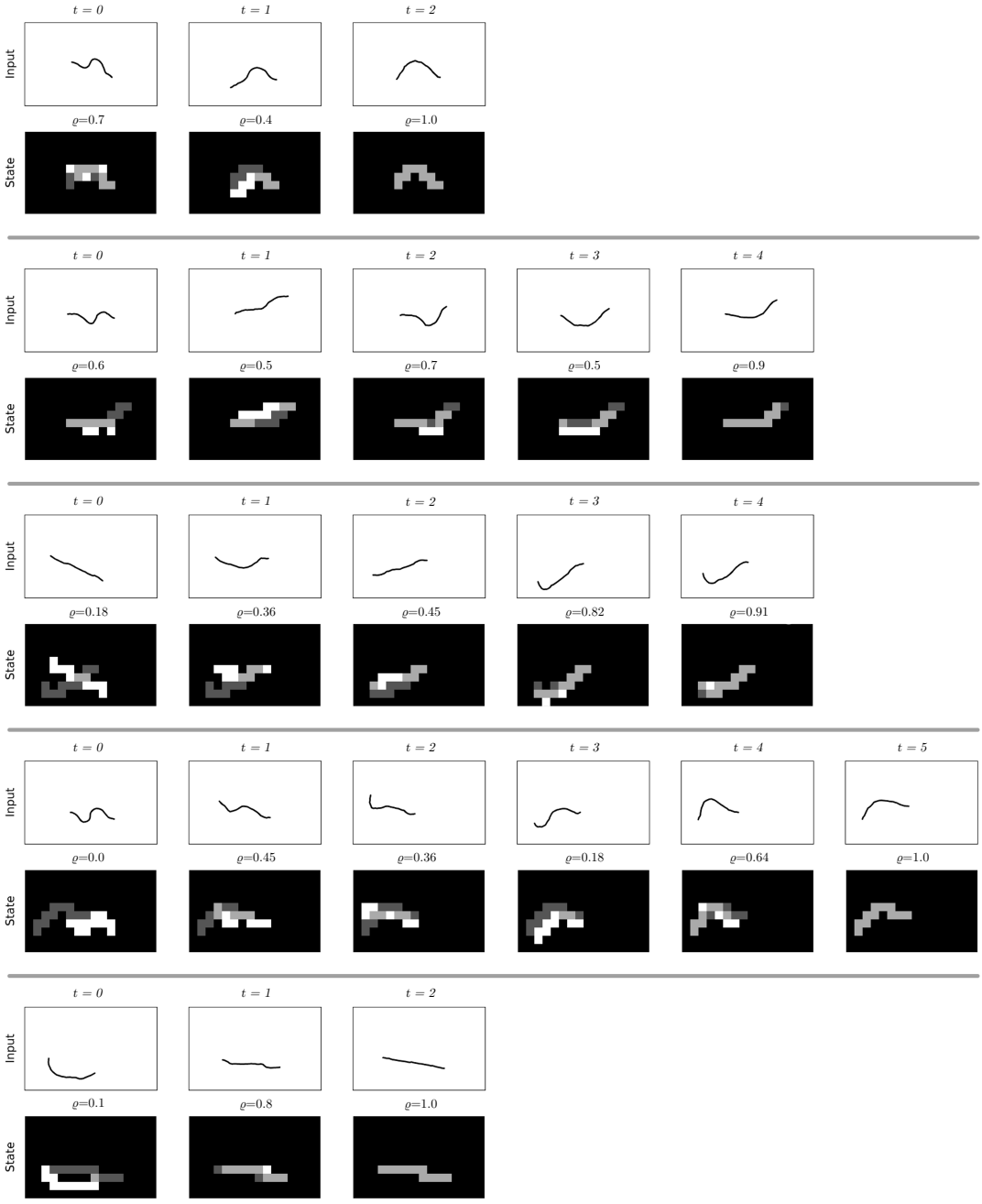


FIGURE 5.8: Second set of 5 experiments that shows the DLO deformation steps performed by the robot using the proposed method. The images are binarized for visual clearance. The final shape corresponds to an overlap score greater than 90% ($\varrho(s_t) >= 0.9$). The state cells are: black if $s_t^{i,j} = 0$; white if $s_t^{i,j} = 2$; light-grey if $s_t^{i,j} = 3$; dark-grey if $s_t^{i,j} = 1$.

Chapter 6

Image Segmentation on Auto-generated Training Datasets

As stated multiple times in this thesis, the presence of a reliable sensing system is a necessary condition to successfully manipulate a DLO, but often it is also one of the biggest challenge in the task development. In this chapter we investigate the image segmentation of electric wires with the dual objective of simplifying the process of training dataset generation and maximizing the quality of the predictions. The study here exposed is preparatory to the development of an algorithm, named Ariadne+, that is still in progress and and aims to improve the reliability and the velocity of the system presented in chapter 5.

6.1 Introduction

The availability of big public datasets [125, 126, 67] has promoted advances of deep learning algorithms in computer vision applications, such as image classification, object detection and semantic segmentation. Thus, the key issue in modern computer vision deals more and more with gathering and labeling big amounts of data. Usually, the process of segmenting and annotating the training images is performed manually, and it is notoriously tedious, inaccurate and time consuming. Moreover, the more complex the visual perception task is, the slower becomes the required annotation procedure. For instance, labeling a single image for 2D semantic segmentation can take several hours per image. Innovative companies, like Scale.ai[1], Superannotate.ai[2], Segments.ai[3] and many others, are basing their business on advanced image labeling pipeline that can speed-up and lighten the burden. These solutions often exploit a superpixel algorithm which helps the user to quickly select large portions of the image instead of individual pixels. Other new approaches rely on weakly supervised learning [127] as Segments.ai that iterates between image labeling and model training in order to provide the user with initial – coarse – labels for each new image instead of having it labeled from scratch.

The aforementioned big public datasets [125, 126, 67] usually concern general classes (e.g. person, car, tree, cat, dog, etc.) that may not suit the needs of a specific task. Robotic applications, especially in industrial settings, typically require the detection or segmentation with very high success rate of small but very specific set of object instances captured from different viewpoints in highly-cluttered scenes. Electric wires, more than other objects, have some peculiarities that bring to some interesting challenges on segmentation tasks: 1) they are deformable objects, which

[1] https://scale.ai
[2] https://superannotate.ai
[3] https://segments.ai

means that they are not characterized by a specific shape; 2) they are very lacking in features; 3) they aren't characterized by any particular color. Since a cable can feature a wide variety of shapes and colors, to train a segmentation model, the generation of a large scale dataset to cover such great variability is necessary.

This work is motivated by the lack of simple and effective solutions to generate big image dataset for training, specifically in the field of cable-like objects. We present here a method, introduced in [128], which relies on the chroma-key technique and enables to easily label a given object on an entire video sequence. Image segmentation is a key point to address sub-tasks like cable grasp, terminal insertion and wire routing. Therefore, in this work we focus mainly on electric wires, even thought we show also the applicability of the proposed method to other object typologies. To generate large datasets, a novel labeling pipeline demanding a minimal human intervention despite the volume of produced labeled data is implemented. First, a video sequence of the target object is taken with a proper background which should be homogeneous and easy to be distinguished from the target. Then, the user does not have to manually label the acquired images, but, instead: a) he/she has simply to tune 1 (possibly 3) parameter once per video sequence; b) the target object will be automatically segmented in the entire sequence, by producing a superimposed pixels mask for each frame, by exploiting chroma key (a well known technique used to compose two images); c) the original video sequence backgrounds can, therefore, be replaced to increase the domain randomization. The main contributions of this work can be summarized as follows:

- The first chroma key approach for data labeling;
- An easy and reliable procedure to automatically generate large training datasets of specific items for semantic segmentation;
- A public high-quality dataset[4] of electric wires for semantic segmentation in general purpose applications;
- Tests and comparisons of different state-of-the-art algorithms on this dataset.

6.2 Related Works

The annotation processes for semantic segmentation is labor-intensive using traditional methods [129, 67]. A lot of research effort have been spent on investigating alternative strategies to help the human operator in this task [130]. Advanced solutions like weakly or semi-supervised segmentation have been proposed.

Weakly supervised learning studies attempt to construct predictive models by learning with incomplete, inexact or inaccurate supervision [127]. Weakly supervised learning for semantic segmentation employs different levels of supervision, like labeling only few pixels (e.g. interactive methods [131]), grouping images containing common objects (e.g. co-segmentation [132]) or providing only image-level labels [133]. In interactive segmentation frameworks [131] small portions of target objects are roughly highlighted by human operators through markers, called seeds. These seeds are used for a training stage that will produce some rough labels for all other images. The user can then produce more seeds and repeat the procedure until the desired quality level is reached.

Object co-segmentation [132] aims to detect and segment the semantically similar objects from a set of images. It gives very weak prior that the images contain

[4] https://www.kaggle.com/zanellar/electric-wires-image-segmentation

the same objects for automatic object segmentation. Although there is a certain gap between models trained by weak/semi-supervision and models trained by full supervision, many researchers are making efforts to reduce the gap.

Several approaches for creating datasets have been developed also within the robotics research community. A semi-automatic method to create labeled datasets for object detection is presented in [134]. The system leverages on moving a 2D camera by means of a robot and an augmented reality pen to define initial object bounding box. Zeng et al. [135] present a 6D pose estimation system for Amazon Picking Challenge, where they segment and label the set of target objects placed on the shelf from depth and multi-view information. This work, not only requires depth information, but is also strongly tailor-made on the task's domain. Besides the robotic community, another popular approach to speed-up creation of training datasets consists in the use of synthetic rendered images [136, 86, 137, 138, 139]. However, obtaining a dataset of realistic images requires hours of highly specialized human work to design suitable synthetic scenes along with high-performance graphical hardware. In these cases, a non-photorealistic scene (*i.e.* a simple CAD model rendered on random background) can cause a well-known problem called *domain shift* [140]. In order to reduce this shift, and avoid spending time on photorealism, several *domain adaptation* techniques are applied [141, 142, 143]. Recent works [141, 144, 145] focus on developing ad-hoc adaptation techniques to close the performance gap between training and test distribution. Unfortunately, the performance achievable is still quite far from those obtainable training on real data or fine-tuning on few annotated samples.

In this work we propose a method to automatically create a training dataset for semantic segmentation from real images and we validate it on electric wires by training different segmentation algorithms. To the best of our knowledge this is the first public dataset for semantic wire segmentation, moreover we are the first presenting this method for generating high-quality datasets from real images with minimal human intervention.

Visual perception of Deformable Linear Objects (DLOs), e.g. wires, cables, ropes, etc., has been typically addressed in fairly simple settings. In [38] Augmented Reality markers are deployed to track end-points, while in other works, like [146], detection relies on background removal. Yan at al. [16] developed a more sophisticated method that relies on Gaussian Mixture Models, but it requires the assumption of having a good color contrast between object and background (which has to be homogeneous). The state-of-the-art solution for DLOs detection is presented in [10]. This algorithm, called Ariadne, is based on biased random walks over the Region Adjacency Graph built on a super-pixel over-segmentation of the source image. Unfortunately, this approach has some weakness: it requires an external detector to localize cable terminals; the prediction is intrinsically quite slow due to the exploration process; it can easily fail due to perspective effects or when cables are adjacent.

6.3 Automatic Dataset Generation

In section 6.1 we underlined the importance of a smart solution to collect training data for data-driven models that requires less human intervention possible. In this section, we detail our method and we present a dataset generated for semantic segmentation of electric wires. The proposed strategy employs chroma key to firstly label a set of images and then replacing the background to randomize the domain and enlarge the dataset.

6.3.1 Auto-labeling with Chroma Key

The Chroma Key (CK) is a technique widely used in film and motion picture industries to combine two images together (usually foreground and background). It requires a foreground image I_{fg} containing a target object that we want to overlap to a background image I_{bg}. The target must be placed in front of a monochromatic panel, called screen (usually green or blue). The technique consists of a *chroma-separation* phase, where we isolate the target object (foreground) from the monochromatic panel (original background) and then an *image-overlay* phase, where we compose the foreground and a new background. In the *chroma-separation* phase, we choose a specific hue range which contains solely the color of the screen (e.g. green) and exclude any other color belonging to the foreground. Then, by finding the pixels within that range, we obtain a mask for the target I_{mt} and a complementary mask for the monochromatic background I_{ms}. Thus, creating a dataset with this technique is really straightforward and it can be done in 2 steps:

1. record an high quality video of the target object on a green screen, from which we gather the input images;

2. find the chroma range of the pixels belonging to the monochromatic background and create the corespondent mask with *chroma separation*.

In our dataset, while gathering the images, we hold the electric wire by its extremities and we move it within the frame composing different shapes. To generalize more we also change the light setups, the wire color and the number of wire in the scene. From a random video frame we easily find the hue levels for the specific screen color we are using (green or blue). These levels, once found for one image, remain valid for any other image taken with the same light temperature setting and white balance. Hence, known the chroma range of the screen we immediately obtain the mask for the wire from each frame in the video.

6.3.2 Domain Randomization

The labeling procedure with *chroma separation* automatically generates labeled data ready to be used for training, but with a low variability. In fact, in the gathering phase we need to randomize the scene featuring target object in the following aspects: number of instances, color, size, position and shape. Nevertheless, the background is always uniform and monochromatic. The performance of a segmentation algorithm trained with images in homogeneous backgrounds would be significantly degraded when working in a complex and chaotic environment. Clutter background in fact easily confuses the algorithm, due to possible similarities between the target and the background, especially if it has never seen them in training. This weakness can be readily overcome by replacing the background in the input images (*image-overlay* phase). In fact, by using the masks, we can combine the foreground with a random background that replaces the green screen. This process, known as domain randomization [143, 86], aims to provide enough synthetic variability in training data such that at test time the model is able to generalize to real-world data. Hence, the choice of the background images is a key point for generalizing well to multiple real-world target domains without the need of accessing any target scenario data in training.

The backgrounds that we propose for a domain-independent dataset can be divided in 3 categories: (1) lowly textured images with shadows and lights; (2) highly textured images with color gradients and regular or geometric shapes; (3) highly

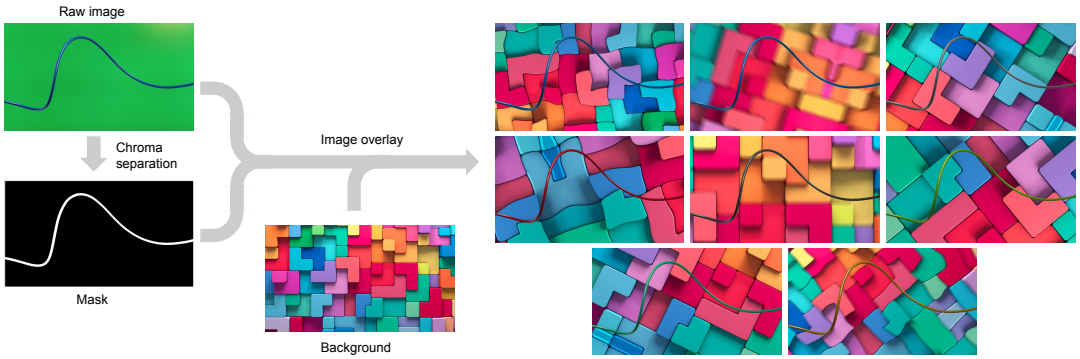


FIGURE 6.1: Schematic process to generate the 8 synthetic images by background-foreground separated augmentation and *image-overlay*.

textured images with chaotic and irregular shapes. These backgrounds introduce high variance in the environment properties that should be ignored in the learning task. For instance, in our task the segmentation algorithm will ignore shadows and cubic or spherical objects, while it should focus more in cylindrical shapes, hence we chose the set of backgrounds in Figure 6.2 according to these considerations.

The presented method introduces two main difficulties that must be faced. The first evident issues of CK concerns the color of the target object. In fact, the color histogram of the object should be well far enough to the range reserved for the screen, or in other words we can not have green wires on a green screen. This implies that the segmentation algorithm never sees green wires in training, thus if it encounters a green wire in a real scene, it would likely produce some false negative. The solution to this issue are two: we can use a different background for the green objects (e.g. a blue screen) or, as we actually do in our dataset, we can randomize the hue of the wire trying to cover also the missing color range (i.e. green). Another issues is caused by the background replacement, which introduces a discontinuity in the synthetic image generated. This may be problematic for the learning, especially in our case with the wires, since the algorithm will probably focus on that sharp feature to segment the object, compromising the prediction in a real image, devoid of the learnt discontinuity. To overcome this issue, the output image I_{out} is obtained according to the following formula

$$I_{\text{out}} = I^{\mathcal{G}}_{\text{mt}} I_{\text{fg}} + (1_{h\times w} - I^{\mathcal{G}}_{\text{mt}}) I_{\text{bg}}. \tag{6.1}$$

i.e. as a linear combination of the foreground I_{fg} and background I_{bg} images weighted respectively by the target mask processed by a Gaussian filter $I^{\mathcal{G}}_{\text{mt}} = \mathcal{G}(I_{\text{mt}})$ and its complement $(1_{h\times w} - I^{\mathcal{G}}_{\text{mt}})$, where $1_{h\times w}$ is a unit matrix with the same size of the mask.

6.3.3 Electric Wire Dataset

The strategy presented in this section has been employed to generate a dataset of 28584 RGB images 720×1280 for semantic segmentation of electric wires. The raw dataset has 3176 images and it includes blue, red, yellow, white and black wires, with different light setups and shapes.To improve the screen and wire separation, besides the hue, we also use the saturation and value channels. For each raw image, a background image (4000×2248) is randomly picked among the 15 shown in Figure 6.2

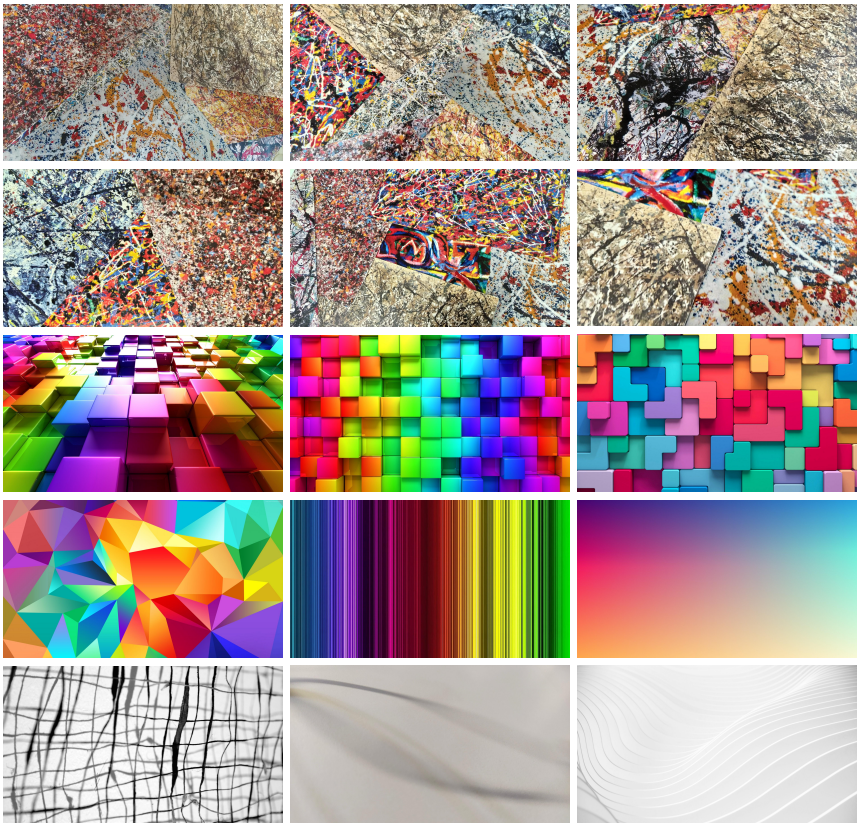


FIGURE 6.2: Images used to replace the background in the output dataset.

and 8 new synthetic images are created, as visible in Figure 6.1. In each new image, foreground and background are separately augmented (by using the mask) before the merging. In particular, the background is randomly flipped, shifted, scaled and rotated (all with probability $p = 0.5$). Then, it is processed with motion blur and elastic transformation ($p = 0.2$), and in the end it is randomly cropped at 1280×720 ($p = 1$). The foreground, instead, is transformed only by shuffling the channels ($p = 0.5$), converting to grey ($p = 0.1$) and randomizing the hue in the range of $[-100, 100]$ ($p = 0.5$).

6.3.4 Final Considerations on the Output Dataset

The dataset produced by our method contains mainly synthetic images produced by chroma-key overlay. However, the reality gap in the resulting dataset is considerably small compared to those that might be obtained from rendering or simulation. In fact, the main visual discrepancy between real and output images is the object's contour, which has already been smoothed with the combination in Equation 6.1. To further reduce the gap, we add to the dataset also the input images with the original background, and to avoid over-fitting on green background we randomize the hue and shuffle the channels in training.

The types of background that we suggest to use are intended to make the dataset general purpose and domain-independent. In fact, in the next section we are going to experimentally validate our wires dataset on several scenarios, empirically proving that a set of abstract backgrounds is sufficient to obtain highly satisfactory predictions also in real environments never seen in training. Clearly, to improve the

results on a specific real domain, fine-tuning can be also performed. We point out that there are actually two ways to do so. The first is the traditional fine-tuning on a small set of manually labeled images. The second consists in creating a dataset using photos of the specific task environment as backgrounds.

6.4 Semantic Segmentation

Two deep learning networks are exploited to perform the training and testing needed to validate our work, namely DeeplabV3+ [147] and HRNet [148].

DeepLabV3+ [147] is an encoder-decoder network which is at the state of the art in deep learning semantic segmentation. It is the last iteration of the famous DeepLab family models. It employes the encoder-decoder structure combined with atrous spacial pyramid pooling (ASPP). As encoder module, DeepLabV3 is used. It is able to encode multi-scale contextual information. The presence of atrous convolutions, instead of the common convolutions, allows the explicitly control the resolution of features computed (via the output stride parameter). For the semantic segmentation task, an output stride of 16 (or 8) is used for denser features. Concerning the decoder, it consists of a simple yet effective module which refines the segmentation results along object boundaries. Here, the low-level features are concatenated to the bi-linearly upsampled (4x) high-level features coming from the decoder. Several convolutions are performed to refine the features and a final upsampling (4x) is performed. Compared to a straightforward one bi-linearly 16x upsampling, the presented decoder module performs much better.

High-Resolution Network (HRNet) [148] is the state of the art in diverse fields such as human pose estimation, semantic segmentation and object detection. It maintains high-resolution representations through the whole network layers by connecting the high-to-low resolution convolution streams in parallel and by repeatedly exchange the information across resolutions. The benefit of such structure consists in having a representation semantically richer and spatially more precise.

6.4.1 Training and Test

We train DeepLabV3+ with a ResNet-101 backbone for 200 epochs, with batch size 10, output stride 16, separable convolutions, using Adam for the optimization and employing a polynomial learning rate adjustment policy starting from 10^{-6} to a minimum of 10^{-9}, with power 0.95. HRNet is instead initialized with a pretrained model on ImageNet. The network is then trained for 270 epochs, with batch size 6, using SGD for the optimization, weight decay 5×10^{-4}, momentum 0.9, initial learning rate of 1×10^{-5}. The learning rate adjustment policy is polynomial with a power of 0.9. The early stopping in both the training is configured to end the process when the validation loss does not decrease for 5 epochs in a row. Both the models are implemented in PyTorch 1.4.0 and trained with an NVIDIA GeForce GTX 2080 Ti on an Intel Core i9-9900K CPU clocked at 3.60GHz. The data augmentation scheme include hue randomization, channel shuffling, flipping and finally resizing (360×640).

The training dataset is obtained from 90% of the original dataset auto-generated as in section 6.3, while the validation is done on the remaining 10%. To test the algorithms, we use another dataset of 60 manually labeled images collected in different real scenarios. The test dataset is composed by 4 categories of 15 images each:

C1: scenes with only the target wires laying on a surface and no other disturbing objects. The difficulties in this scenes are the high contrast shadows of the

Algorithm	*C1*	*C2*	*C3*	*C4*	Tot.
DeepLabV3+	0.928	0.934	0.943	0.935	**0.935**
HRNet	0.923	0.939	0.911	0.926	**0.925**
Ariadne	0.655	0.512	0.632	0.595	0.598

TABLE 6.1: The average Dice Coefficient computed for each algorithm, across the images of each test set (*C1*, *C3*, *C4*, *C2*) and the union (*Tot*). In all the tests the predictions are thresholded at 0.5.

wires, possible chroma similarities between the wires and the background, the dense crosses of wires, the light settings and the perspective distortions.

C2: scenes with the target wires only on a highly-featured and complex background and no other disturbing objects. Here the challenge for the algorithms is to extract the correct features belonging to the wires in a cluttered scene.

C3: scenes with the target wires in a realistic industrial setting like an electric panel. These can be considered as an example of an application setting, where the difficulties may be given by metallic surface reflecting the wires and other disturbing objects like commercial electromechanical components characteristic of these panels.

C4: scenes with the target wires in other generic realistic settings among other objects of different nature. The difficulties in these scenes are several and a combination of those found before.

Each algorithm produces a mask M_p which corresponds to the the predicted semantic segmentation of the wire. We evaluate and compare the outputs by means of the Dice coefficient $\text{Dice} = 2\frac{|M_p \cap M_{gt}|}{|M_p|+|M_{gt}|}$, where M_{gt} is the ground truth. Table 6.1 resumes the average Dice obtained in the test dataset by DeepLabV3+, HRNet and Ariadne [10], state-of-the-art algorithm for DLO segmentation. Ariadne yields a b-spline model for each wire which is here used as predicted mask M_p. In order to make the the comparison with Ariadne more meaningful, we tuned the parameters specifically for the given test dataset and we manually found for each wire the b-spline thickness best fitting with the target. In Figure 6.4 are visible few example of test images for each category and the outputs of both DeepLabV3+ and HRNet, where true positive ($M_p \cap M_{gt}$), false positive ($M_p - M_{gt}$) and false negative ($M_{gt} - M_p$) are shown in yellow, red and green respectively.

From these tests we can conclude that the auto-generated dataset reaches an high level of reliability ($Dice > 0.9$) for both HRNet and DeepLabV3+ in any scenario without any fine-tuning. More in detail, with reference to Figure 6.4, we can observe that prospective distortions (*C1*-Sample5, *C2*-Sample 4), color similarities with the background (*C1*-Sample3, *C1*-Sample4), multiple-wire dense intersections (*C1*-Sample2, *C2*-Sample5), wire reflections in metallic surfaces (*C3*-Sample3) and strong shadows (*C1*-Sample2) are all correctly solved using both the algorithms. Moreover, the hue randomization trick used in the foreground images enables the algorithms to correctly recognize also green wires (*C3*-Sample4); whereas the selection of background images allows to effectively segment electric wires in settings never seen in training, very confusing and also with other objects that might look similar to them, such as the table border in *C4*-Sample5 or the handle of the pliers in *C4*-Sample1.

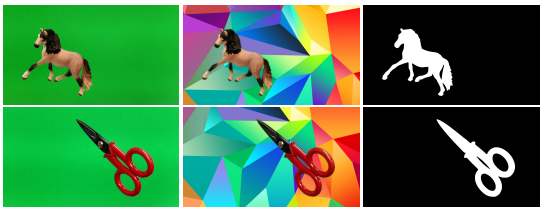


FIGURE 6.3: Sample images from other hypothetical dataset auto-generated with the proposed CK-based technique.

The quantitative results of Table 6.1 show that DeepLabV3+ performs on average slightly better than HRNet. In fact, from the qualitative comparison of Figure 6.4, we observe that the predictions of HRNet are on average a little less confident and sharp at the edges (*C1*-Sample2, *C1*-Sample3). This might be due to an higher sensitivity of HRNet to the reality gap, already discussed in subsection 6.3.2, that we tried to reduce by introducing the Gaussian blur on the mask in Equation 6.1. However, even thought the predictions of DeepLabv3+ are more precise, it produces evident false negative (like those in *C2*-Sample3) more frequently than HRNet. Table 6.1 revels also that both DeepLabV3+ and HRNet trained on our dataset obtain significantly higher performance than the baseline Ariadne.

6.5 Conclusions

In this work, we address the problem of recognising and segmenting electric wires from images, which are deformable objects very common in many applications but also lacking of visual features. A novel strategy to automatically generate a domain-independent dataset has been presented and experimentally validated by training and testing two algorithms, namely HRNet and DeepLabv3+. The experimental results show the effectiveness of the dataset, that enables the segmentation algorithms to correctly recognize the wires in different settings never seen in training with an Average Dice index greater than 0.92. We underline that the presented approach to create the electric-wire dataset can be applied to any other object small enough to be placed and moved in front of a monochromatic panel, like those in Figure 6.3. In future works, we will formally extend this method to generic objects, then we will also further reduce the human intervention in the *chroma-separation* phase by employing a learning-based methods and improve the *image-composition* phase by reducing the reality gap in the edges.

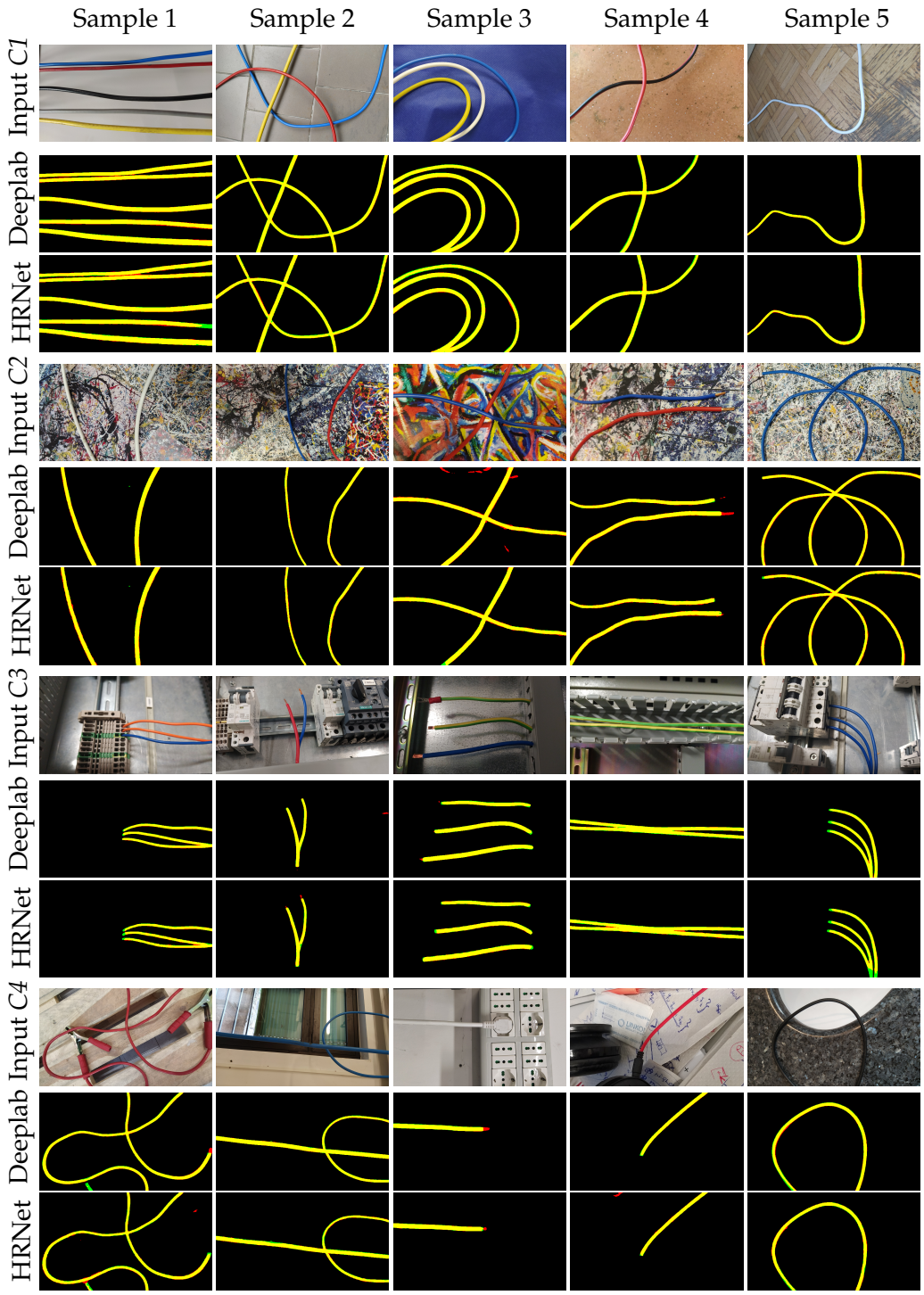


FIGURE 6.4: Qualitative evaluation of DeepLabV3+ and HRNet using 20 sample images from the test set (5 images from each category). The yellow areas are the true positives, the red areas the false positive and green areas the false negative.

Chapter 7

Conclusions and Future Works

Robotic applications that deal with DLOs have usually to face two main challenges: the insertion of the DLO in a hole and the control of its shape. In this thesis we investigated both these tasks, exploiting tactile and visual sensors embedded with suitable deep-learning techniques. Due to the complex dynamics of DLOs and their infinite amount of possible state configurations, classical model-based approaches turn out to be ineffective, since they oblige the search for a compromise between very slow computation and poor reliability. For these reasons and thanks to huge advances in deep learning, the robotics community has now almost entirely shifted to using these algorithms to tackle difficult problems such as manipulating deformable objects. In particular, the effectiveness of learning-based algorithms for sensors data analysis is undeniable, in fact they are able to quickly extract useful and reliable information, that can be later used for the manipulation. Inside the works presented in this thesis we employed different learning-based algorithms for several purposes: In chapter 2 we employed YOLO [49] (a CNN-based object detector) to locate the wire terminal on the table, while we trained and compared different machine learning algorithms for regression and classification (i.e. MLP, SVM, RF) to evaluate the grasp and insertion of the wire; In chapter 3 we developed a novel approach to estimate the bi-dimensional pose of a target object using YOLOv3 [81] Object Detector (providing also a fast and reliable labeling method); In chapter 4, a RNN with LSTM cells maps the signals from a tactile sensor into an estimate of the force vector exerted on the grasped wire, in order to guide the insertion into a hole; In chapter 5, a decision making process uses a DQN agent with a CNN-based critic for predicting the optimal grasping point on a DLO in order to achieve a target deformation; In chapter 6, DeepLabV3+ [147] and HRNet [148] (two CNN-based image segmentation algorithms) are trained on an auto-generated dataset of electric wires and compared.

The main contributions introduced with the research activities that compose this thesis are: 1) a pipeline for inserting a DLO in a hole that starts by grasping the extremity from an unknown pose on the table; 2) a novel approach to guide the blind insertion of a DLO into a hole by analyzing only the tactile feedback from a fingers-mounted sensor; 3) a robot learning-based system for autonomous deformation of a rope from/to a general shape using visual feedback capable to work with any cluttered backgrounds. Both these contributions are enhanced by two dedicated methods for generating large training dataset with minimum human intervention. To conclude, let us point out that bringing together all the contributions presented in the thesis, a preliminary development of an automatic wiring system naturally emerges. In fact, reminding the wiring process outlined in the Introduction: the connection of the wires-terminals with the electromechanical components can be addressed using the results of chapter 2, chapter 3, chapter 4, while the detection and rearrangement of the wires was investigated in chapter 6, chapter 5.

7.1 Current and Future Works

In this section we introduce two works currently in progress. The first is a novel algorithm that uses the results of chapter 6 to estimate the shaped of electric wired as a b-spline models. The second is a pipeline that extends the work presented in chapter 5 to more generic DLOs which might have also elastoplastic properties. Further improvements can also be done to each work presented in the thesis in terms of speed and reliability, few of these are reported in the corresponding chapters. Finally, other futures works can be concerning the extension of all the works to three-dimensional sensing and reshaping of DLOs.

7.1.1 Instance Image Segmentation and Shape Estimation of DLOs

In chapter 6 we investigated the image segmentation of electric wires and we trained two state-of-the-art algorithms (i.e. DeepLabV3+ and HRNet) on an auto-generated dataset. Using the results of this work we are able to efficacy recognize the wires in an image, or in other words we can find the sets of pixels in the image belonging to the target objects. However, this information might not be sufficient for a robotic manipulation task, like the one represented in chapter 5, since it does not carry an explicit shape estimation of the DLO. In fact, the algorithm employed in subsection 5.5.2 for DLOs detection (namely *Ariadne* [10]), yields a b-spline model of the detected object alongside a segmentation of the whole image. Unfortunately, Ariadne has few drawbacks: the computations are highly time consuming (1382*ms* for the initialization and 2506*ms* for the discovery phase); the results are often insufficiently reliable; it requires a tedious manual tuning of a large set of parameters.

In a current work we are developing a novel method that combines a semantic segmentation algorithm like DeepLabV3+ with the core idea of Ariadne, we call this algorithm *Ariadne+*. The goal of Ariadne+ is to obtain the instance segmentation masks and the shape estimation (b-splines) of all the electric-wires in a given RGB image. Ariadne+ aims not only to significantly improve the time complexity, the reliability and the precision of its predecessor Ariadne, but also to reduce the parameters that requires manual tuning.

The proposed solution is based on a Region Adjacency Graph, built on a masked superpixel segmentation of the input image, which has a node for each superpixel and each node is connected with an edge to all the nodes that correspond to neighboring superpixels. Generally, speaking a superpixel segmentation consists in partitioning images into local meaningful subregions by capturing local similarity among the pixels. Whereas, a masked superpixel segmentation [149] is able to perform the clustering only within regions of interest given by a binary mask. Hence, to our purpose, we use one of the two image segmentation algorithms, i.e. DeepLabV3+, trained in chapter 6 for detecting electric wires. The Region Adjacency Graph build in this way, allows us to effectively find the b-spline models of all the wires in the input image as different paths on the graph (as was done in Ariadne). In fact, once detected the wires with a image segmentation algorithm, the entire procedure that identifies each wire is based on simple and fast computations on the graph (or to the correspondent superpixels). The success of Ariadne+ against Ariadne, is mainly due to the strategic employment of few learning-based solutions. In fact, the graph build on the wire detection provided by DeepLabV3+ is significantly simpler than the one built on the raw image (as done in Ariadne). This makes the path searching process faster and more precise, since along a single wire without crosses each node has only 2 neighbors. In this settings, the only difficulty is encountered in the cross

points, where the algorithm must choose the correct path continuation. Differently to Ariadne, in Ariadne+ we use a CNN with triplet loss to predict the next node. This CNN provides a "similarity score" among the image patches corresponding the nodes around the cross point. Finally, another CNN is trained to classify which wire in on the top in a cross. Preliminary experiments, ran on a test set of images with cluttered background and multiple crosses, show a Average Precision of 0.78 for Ariadne+ and 0.32 for Ariadne, while the mean execution time is 560ms for the former and 3800ms for the latter.

In summary, the main contributions of this work will be: 1) a reliable and time effective complete approach for the instance segmentation of wires in real scenarios; 2) modeling of the detected wires in terms of b-splines; 3) an open source implementation of the entire algorithm; 4) a comprehensive experimental validation in terms of segmentation capabilities and timings performances of the approach; 5) a comparison against standard Ariadne, the current state-of-the-art segmentation algorithm for DLOs.

7.1.2 Dual-arm Manipulation of Generic DLO with Elastoplastic Properties

In chapter chapter 5 we addressed the shape control of a highly deformable object laying on a table, using a single arm robot and a sequence of pick-and-drop actions. Note that, in those settings the target DLO can be considered purely plastic, since it holded the given shape. Hence, another research activity currently in progress aims to extend the manipulation capabilities of the reshaping system presented in chapter 5 to a generic elastoplastic DLO. In this settings, where elasticity is introduced and DLO stiffness can be increased, the system setup previously suggested need to be modified. In fact, a dual-arm robot with a new sequence of motion primitives resulting from the synergistic combination of the two arms would provide a better solution. A straight forward approach might be a sequence of *hold-and-move* operations, where an arm holds the DLO in a point while the other grasps and moves another point in a target releasing location. Here, the deformation trajectory must be consistent with the segment of DLO between the two grippers, to avoid excessive tension that can compromise the grasp. In other words we can state that, the euclidean distance between the holding point and the target one must be less then the arc distance along the DLO between the two grasped points. In order to manipulate the DLO, we need an estimate of the its shape as a keypoints path. Hence, both the algorithms *Ariadne* and it extension *Ariadne+* (discussed in the previous subsection) can be exploited for the purpose. When the DLO has elastoplastic properties, the releasing point should also be chosen correctly. In fact, the system need to be capable of predicting the elastic deformation occurring in the object after it opens the gripper. A learning-based method can be trained in self-supervised learning or using reinforcement learning, by autonomously interacting with the DLO. Using the shape information provided by Ariadne or Ariadne+, the algorithm can easily learn to map the initial and target shapes into a optimal deformation.

Bibliography

[1] Steven Kinio and Alexandru Patriciu. "A comparative study of H and PID control for indirect deformable object manipulation". In: *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. 2012, pp. 414–420.

[2] Antoine Petit, Fanny Ficuciello, Giuseppe Andrea Fontanelli, Luigi Villani, and Bruno Siciliano. "Using physical modeling and RGB-D registration for contact force sensing on deformable objects". In: *ICINCO 2017-14th International Conference on Informatics in Control, Automation and Robotics*. Vol. 2. 978-989-758-263-9. ScitePress; Springer. 2017, pp. 24–33.

[3] Olivier Roussel and Michel Taıöx. "Deformable linear object manipulation planning with contacts". In: *Robot Manipulation: What has been achieved and what remains to be done? Full day workshop at IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2014.

[4] Yinxiao Li, Yan Wang, Yonghao Yue, Danfei Xu, Michael Case, Shih-Fu Chang, Eitan Grinspun, and Peter K Allen. "Model-driven feedforward prediction for manipulation of deformable objects". In: *IEEE Transactions on Automation Science and Engineering* 15.4 (2018), pp. 1621–1638.

[5] Daniel Kruse, Richard J Radke, and John T Wen. "Collaborative human-robot manipulation of highly deformable materials". In: *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2015, pp. 3782–3787.

[6] Angel Delgado, Carlos A Jara, and Fernando Torres. "Adaptive tactile control for in-hand manipulation tasks of deformable objects". In: *The International Journal of Advanced Manufacturing Technology* 91.9-12 (2017), pp. 4127–4140.

[7] Daniele De Gregorio, Riccardo Zanella, Gianluca Palli, Salvatore Pirozzi, and Claudio Melchiorri. "Integration of robotic vision and tactile sensing for wire-terminal insertion tasks". In: *IEEE Transactions on Automation Science and Engineering* 16.2 (2018), pp. 585–598.

[8] Riccardo Zanella, Daniele De Gregorio, Salvatore Pirozzi, and Gianluca Palli. "DLO-in-Hole for Assembly Tasks with Tactile Feedback and LSTM Networks". In: *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE. 2019, pp. 285–290.

[9] Alessio Caporali and Gianluca Palli. "Pointcloud-based Identification of Optimal Grasping Poses for Cloth-like Deformable Objects". In: *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Vol. 1. IEEE. 2020, pp. 581–586.

[10] Daniele De Gregorio, Gianluca Palli, and Luigi Di Stefano. "Let's Take a Walk on Superpixels Graphs: Deformable Linear Objects Segmentation and Model Estimation". In: *Asian Conference on Computer Vision*. Springer. 2018, pp. 662–677.

[11] Yoshihisa Tsurumine, Yunduan Cui, Eiji Uchibe, and Takamitsu Matsubara. "Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation". In: *Robotics and Autonomous Systems* 112 (2019), pp. 72–83.

[12] Shaowei Cui, Rui Wang, Junhang Wei, Fanrong Li, and Shuo Wang. "Grasp state assessment of deformable objects using visual-tactile fusion perception". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 538–544.

[13] Jianing Qian, Thomas Weng, Luxin Zhang, Brian Okorn, and David Held. "Cloth Region Segmentation for Robust Grasp Selection". In: *arXiv preprint arXiv:2008.05626* (2020).

[14] Daniele De Gregorio, Riccardo Zanella, Gianluca Palli, and Luigi Di Stefano. "Effective Deployment of CNNs for 3DoF Pose Estimation and Grasping in Industrial Settings". In: *25th International Conference on Pattern Recognition*. 2020.

[15] Yu Song, Kang Yang, Xin Jiang, and Yunhui Liu. "Vision Based Topological State Recognition for Deformable Linear Object Untangling Conducted in Unknown Background". In: *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. 2019, pp. 790–795.

[16] Mengyuan Yan, Yilin Zhu, Ning Jin, and Jeannette Bohg. "Self-Supervised Learning of State Estimation for Manipulating Deformable Linear Objects". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 2372–2379.

[17] P Jiménez. "Survey on model-based manipulation planning of deformable objects". In: *Robotics and computer-integrated manufacturing* 28.2 (2012), pp. 154–163.

[18] Shigang Yue and Dominik Henrich. "Manipulating deformable linear objects: sensor-based fast manipulation during vibration". In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*. Vol. 3. IEEE. 2002, pp. 2467–2472.

[19] Alex X Lee, Abhishek Gupta, Henry Lu, Sergey Levine, and Pieter Abbeel. "Learning from multiple demonstrations using trajectory-aware non-rigid registration with applications to deformable object manipulation". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 5265–5272.

[20] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. "Combining self-supervised learning and imitation for vision-based rope manipulation". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 2146–2153.

[21] Priya Sundaresan, Jennifer Grannen, Brijen Thananjeyan, Ashwin Balakrishna, Michael Laskey, Kevin Stone, Joseph E Gonzalez, and Ken Goldberg. "Learning Rope Manipulation Policies Using Dense Object Descriptors Trained on Synthetic Depth Data". In: *arXiv preprint arXiv:2003.01835* (2020).

[22] Wilson Yan, Ashwin Vangipuram, Pieter Abbeel, and Lerrel Pinto. "Learning Predictive Representations for Deformable Objects Using Contrastive Estimation". In: *arXiv preprint arXiv:2003.05436* (2020).

[23] Haifeng Han, Gavin Paul, and Takamitsu Matsubara. "Model-based reinforcement learning approach for deformable linear object manipulation". In: *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*. IEEE. 2017, pp. 750–755.

[24] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. "Learning to manipulate deformable objects without demonstrations". In: *arXiv preprint arXiv:1910.13439* (2019).

[25] Rita Laezza and Yiannis Karayiannidis. "Shape Control of Elastoplastic Deformable Linear Objects through Reinforcement Learning". In: ().

[26] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards". In: *arXiv preprint arXiv:1707.08817* (2017).

[27] H. Nakagaki, K. Kitagaki, T. Ogasawara, and H. Tsukune. "Study of insertion task of a flexible wire into a hole by using visual tracking observed by stereo vision". In: *Proc. of the Int. Conf. on Robotics and Automation*. Vol. 4. 1996, pp. 3209–3214.

[28] H. Nakagaki, K. Kitagaki, T. Ogasawara, and H. Tsukune. "Study of deformation and insertion tasks of a flexible wire". In: *Proc. of the Int. Conf. on Robotics and Automation*. Vol. 3. 1997, pp. 2397–2402.

[29] Axel Remde, Dominik Henrich, and Heinz Wörn. "Picking-up deformable linear objects with industrial robots". In: (1999).

[30] Hidefumi Wakamatsu and Shinichi Hirai. "Static Modeling of Linear Object Deformation Based on Differential Geometry". In: *The International Journal of Robotics Research* 23.3 (2004), pp. 293–311.

[31] H. Wakamatsu, K. Takahashi, and S. Hirai. "Dynamic Modeling of Linear Object Deformation based on Differential Geometry Coordinates". In: *Proc. of the Int. Conf. on Robotics and Automation*. 2005, pp. 1028–1033.

[32] M. Moll and L. E. Kavraki. "Path planning for deformable linear objects". In: *IEEE Transactions on Robotics* 22.4 (2006), pp. 625–636.

[33] M. Saha and P. Isto. "Manipulation Planning for Deformable Linear Objects". In: *IEEE Transactions on Robotics* 23.6 (2007), pp. 1141–1150.

[34] A. Theetten, L. Grisoni, C. Andriot, and B. Barsky. "Geometrically exact dynamic splines". In: *Computer-Aided Design* 40.1 (2008), pp. 35–48.

[35] Y. Kataoka and S. Hirai. "Vison-guided individual modeling of bendable cables for their insertion". In: *2010 World Automation Congress*. 2010, pp. 1–8.

[36] J. Schulman, A. Lee, J. Ho, and P. Abbeel. "Tracking deformable objects with point clouds". In: *Proc. of the Int. Conf. on Robotics and Automation*. 2013, pp. 1130–1137.

[37] Kyong-mo Koo, Xin Jiang, Kohei Kikuchi, Atsushi Konno, and Masaru Uchiyama. "Development of a robot car wiring system". In: *2008 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. IEEE. 2008, pp. 862–867.

[38] Xin Jiang, Kyong-mo Koo, Kohei Kikuchi, Atsushi Konno, and Masaru Uchiyama. "Robotized assembly of a wire harness in a car production line". In: *Advanced Robotics* 25.3-4 (2011), pp. 473–489.

[39] Di Guo, Fuchun Sun, Bin Fang, Chao Yang, and Ning Xi. "Robotic grasping using visual and tactile sensing". In: *Information Sciences* 417 (2017), pp. 274–286.

[40] Huaping Liu, Yuanlong Yu, Fuchun Sun, and Jason Gu. "Visual–tactile fusion for object recognition". In: *IEEE Tran. on Automation Science and Engineering* 14.2 (2017), pp. 996–1008.

[41] Oliver Kroemer, Christoph H Lampert, and Jan Peters. "Learning dynamic tactile sensing with robust vision-based training". In: *IEEE Tran. on Robotics* 27.3 (2011), pp. 545–557.

[42] P. Falco, S. Lu, A. Cirillo, C. Natale, S. Pirozzi, and D. Lee. "Cross-modal visuo-tactile object recognition using robotic active exploration". In: *Proc. of the IEEE Int. Conf. on Robotics and Automation*. May 2017, pp. 5273–5280.

[43] Wesley Becari, Luana Ruiz, Bruno GP Evaristo, and Francisco Javier Ramirez-Fernandez. "Comparative analysis of classification algorithms on tactile sensors". In: *Proc. of the Int. Sym. on Consumer Electronics*. 2016, pp. 1–2.

[44] Emil Hyttinen, Danica Kragic, and Renaud Detry. "Learning the tactile signatures of prototypical object parts for robust part-based grasping of novel objects". In: *Proc. of the Int. Conf. on Robotics and Automation*. 2015, pp. 4927–4932.

[45] Adam J Spiers, Minas V Liarokapis, Berk Calli, and Aaron M Dollar. "Single-grasp object classification and feature extraction with simple robot hands and tactile sensors". In: *IEEE transactions on haptics* 9.2 (2016), pp. 207–220.

[46] M. Busi, A. Cirillo, D. De Gregorio, M. Indovini, G. De Maria, C. Melchiorri, C. Natale, G. Palli, and S. Pirozzi. "The WIRES experiment: tools and strategies for robotized switchgear cabling". In: *Proc. of the 27th International Conference on Flexible Automation and Intelligent Manufacturing*. Modena, Italy, 2017.

[47] A. Cirillo, G. De Maria, C. Natale, and S. Pirozzi. "Design and Evaluation of Tactile Sensors for the Estimation of Grasped Wire Shape". In: *Proc. IEEE Int. Conf. on Advanced Intelligent Mechatronics*. Munich, Germany, July 2017, pp. 490–496.

[48] G. De Maria, C. Natale, and S. Pirozzi. "Force/tactile sensor for robotic applications". In: *Sensors and Actuators A: Physical* 175 (2012), pp. 60–72.

[49] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection". In: *Proc. of the Conf. on Computer Vision and Pattern Recognition*. 2016, pp. 779–788.

[50] Chaitanya Mitash, Kostas E Bekris, and Abdeslam Boularias. "A Self-supervised Learning System for Object Detection using Physics Simulation and Multi-view Pose Estimation". In: *arXiv preprint arXiv:1703.03347* (2017).

[51] Georgios Georgakis, Arsalan Mousavian, Alexander C Berg, and Jana Kosecka. "Synthesizing training data for object detection in indoor scenes". In: *arXiv preprint arXiv:1702.07836* (2017).

[52] R. Girshick, J. Donahue, T. Darrell, and J. Malik. "Region-Based Convolutional Networks for Accurate Object Detection and Segmentation". In: *IEEE Tran. on Pattern Analysis and Machine Intelligence* 38.1 (2016), pp. 142–158.

[53] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[54] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[55] Daniele De Gregorio, Federico Tombari, and Luigi Di Stefano. "RobotFusion: Grasping with a Robotic Manipulator via Multi-view Reconstruction". In: *Computer Vision ECCV Workshops*. Springer. 2016, pp. 634–647.

[56] Jon Louis Bentley and Thomas A Ottmann. "Algorithms for reporting and counting geometric intersections". In: *IEEE Tran. on computers* 9 (1979), pp. 643–647.

[57] David Gallup, Jan-Michael Frahm, Philippos Mordohai, and Marc Pollefeys. "Variable baseline/resolution stereo". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE. 2008, pp. 1–8.

[58] R Munoz-Salinas. "ARUCO: a minimal library for Augmented Reality applications based on OpenCv". In: *Universidad de Crdoba* (2012).

[59] Donald J Berndt and James Clifford. "Using dynamic time warping to find patterns in time series." In: *KDD workshop*. Vol. 10. 16. Seattle, WA. 1994, pp. 359–370.

[60] Daniele De Gregorio, Riccardo Zanella, Gianluca Palli, and Luigi Di Stefano. "Effective Deployment of CNNs for 3DoF Pose Estimation and Grasping in Industrial Settings". In: *arXiv preprint arXiv:2012.13210* (2020).

[61] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. "Ssd: Single shot multibox detector". In: *European conf. on computer vision*. Springer. 2016, pp. 21–37.

[62] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in Neural Information Processing Systems*. 2015, pp. 91–99.

[63] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. "SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again". In: *Proc. of the International Conf. on Computer Vision (ICCV), Venice, Italy*. 2017, pp. 22–29.

[64] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. "Real-Time Seamless Single Shot 6D Object Pose Prediction". In: *Proc. of the Conf. on Computer Vision and Pattern Recognition*. IEEE. 2018, pp. 292–301.

[65] Martin Sundermeyer, Zoltan Marton, Maximilian Durner, and Rudolph Triebel. "Implicit 3D Orientation Learning for 6D Object Detection from RGB Images". In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2018, pp. 699–715.

[66] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[67] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. "Microsoft coco: Common objects in context". In: *European Conf. on Computer Vision*. Springer. 2014, pp. 740–755.

[68] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes". In: *International Conf. on Computer Vision (ICCV)*. IEEE. 2011, pp. 858–865.

[69] Tomáš Hodan, Pavel Haluza, Štepán Obdržálek, Jiri Matas, Manolis Lourakis, and Xenophon Zabulis. "T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects". In: *Winter Conf. on Applications of Computer Vision (WACV)*. IEEE. 2017, pp. 880–888.

[70] Yanzhao Zhou, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. "Oriented response networks". In: *Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017, pp. 4961–4970.

[71] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European Conf. on Computer Vision*. Springer. 2014, pp. 818–833.

[72] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. "Spatial transformer networks". In: *Advances in neural information processing systems*. 2015, pp. 2017–2025.

[73] David G Lowe. "Object recognition from local scale-invariant features". In: *Proc. of the International Conf. on Computer vision*. Vol. 2. IEEE. 1999, pp. 1150–1157.

[74] Shaharyar Ahmed Khan Tareen and Zahra Saleem. "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK". In: *International Conf. on Computing, Mathematics and Engineering Technologies (iCoMET)*. IEEE. 2018, pp. 1–10.

[75] Federico Tombari, Alessandro Franchi, and Luigi Di Stefano. "BOLD features to detect texture-less objects". In: *Proc. of the International Conf. on Computer Vision*. IEEE. 2013, pp. 1265–1272.

[76] Jacob Chan, Jimmy Addison Lee, and Qian Kemao. "Border: An oriented rectangles approach to texture-less object recognition". In: *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*. 2016, pp. 2855–2863.

[77] Jacob Chan, Jimmy Addison Lee, and Qian Kemao. "BIND: Binary Integrated Net Descriptors for Texture-less Object Recognition". In: *Proc. CVPR*. 2017.

[78] Bolin Liu, Xiao Shu, and Xiaolin Wu. "Fast Screening Algorithm for Rotation Invariant Template Matching". In: *International Conf. on Image Processing (ICIP)*. IEEE. 2018, pp. 3708–3712.

[79] Farhan Ullah and Shun'ichi Kaneko. "Using orientation codes for rotation-invariant template matching". In: *Pattern recognition* 37.2 (2004), pp. 201–209.

[80] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Pascal Fua, and Nassir Navab. "Dominant orientation templates for real-time detection of texture-less objects". In: (2010).

[81] Joseph Redmon and Ali Farhadi. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[82] Yun Jiang, Stephen Moseson, and Ashutosh Saxena. "Efficient grasping from rgbd images: Learning using a new rectangle representation". In: *International Conf. on Robotics and Automation (ICRA)*. IEEE. 2011, pp. 3304–3311.

[83] Joseph Redmon and Anelia Angelova. "Real-time grasp detection using convolutional neural networks". In: *International Conf. on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 1316–1322.

[84] Fu-Jen Chu, Ruinian Xu, and Patricio A Vela. "Real-World Multiobject, Multigrasp Detection". In: *Robotics and Automation Letters* 3.4 (2018), pp. 3355–3362.

[85] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. "ORB: An efficient alternative to SIFT or SURF". In: *International Conf. on Computer Vision (ICCV)*. IEEE. 2011, pp. 2564–2571.

[86] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *International Conf. on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 23–30.

[87] Anton Milan, Seyed Hamid Rezatofighi, Anthony R Dick, Ian D Reid, and Konrad Schindler. "Online Multi-Target Tracking Using Recurrent Neural Networks." In: *AAAI*. Vol. 2. 2017, p. 4.

[88] Masayuki Inaba and Hirochika Inoue. "Hand eye coordination in rope handling". In: *Journal of the Robotics Society of Japan* 3.6 (1985), pp. 538–547.

[89] Yuan F Zheng, Run Pei, and Chichyang Chen. "Strategies for automatic assembly of deformable objects". In: *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. IEEE. 1991, pp. 2598–2603.

[90] Hirofumi Nakagaki, K Kitagi, Tsukasa Ogasawara, and Hideo Tsukune. "Study of insertion task of a flexible wire into a hole by using visual tracking observed by stereo vision". In: *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*. Vol. 4. IEEE. 1996, pp. 3209–3214.

[91] Hidefumi Wakamatsu, Shinichi Hirai, and Kazuaki Iwata. "Modeling of linear objects considering bend, twist, and extensional deformations". In: *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*. Vol. 1. IEEE. 1995, pp. 433–438.

[92] Hirofumi Nakagaki, Kosei Kitagaki, Tsukasa Ogasawara, and Hideo Tsukune. "Study of deformation and insertion tasks of a flexible wire". In: *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*. Vol. 3. IEEE. 1997, pp. 2397–2402.

[93] Masura Uchiyama and Kosei Kitagaki. "Dynamic force sensing for high-speed robot manipulation using Kalman filtering techniques". In: *Decision and Control, 1989., Proceedings of the 28th IEEE Conference on*. IEEE. 1989, pp. 2147–2152.

[94] J Gamez Garcia, Anders Robertsson, J Gomez Ortega, and Rolf Johansson. "Self-calibrated robotic manipulator force observer". In: *Robotics and Computer-Integrated Manufacturing* 25.2 (2009), pp. 366–378.

[95] Heiko Koch, Alexander König, Alexandra Weigl-Seitz, Karl Kleinmann, and Jozef Suchý. "Force, acceleration and vision sensor fusion for contour following tasks with an industrial robot". In: *Robotic and Sensors Environments (ROSE), 2011 IEEE International Symposium on*. IEEE. 2011, pp. 1–6.

[96] Georg Bätz, Bernhard Weber, Michael Scheint, Dirk Wollherr, and Martin Buss. "Dynamic contact force/torque observer: Sensor fusion for improved interaction control". In: *The International Journal of Robotics Research* 32.4 (2013), pp. 446–457.

[97] Jason Pile, George B Wanna, and Nabil Simaan. "Force-based flexible path plans for robotic electrode insertion". In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE. 2014, pp. 297–303.

[98] Andrea Cirillo, Giuseppe De Maria, Ciro Natale, and Salvatore Pirozzi. "Design and evaluation of tactile sensors for the estimation of grasped wire shape". In: *Advanced Intelligent Mechatronics (AIM), 2017 IEEE International Conference on*. IEEE. 2017, pp. 490–496.

[99] Miao Li, Yasemin Bekiroglu, Danica Kragic, and Aude Billard. "Learning of grasp adaptation through experience and tactile sensing". In: *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. Ieee. 2014, pp. 3339–3346.

[100] Ilya Sutskever, James Martens, and Geoffrey E Hinton. "Generating text with recurrent neural networks". In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011, pp. 1017–1024.

[101] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies". In: (2001).

[102] Sepp Hochreiter and Jurgen Schmidhuber. "Bridging Long Time Lags by Weight Guessing and Long Short Term Memory". In: *Spatiotemporal models in biological and artificial systems* 37 (1996), pp. 65–72.

[103] John E Hopcroft, Joseph K Kearney, and Dean B Krafft. "A case study of flexible object manipulation". In: *The International Journal of Robotics Research* 10.1 (1991), pp. 41–50.

[104] Mark Moll and Lydia E Kavraki. "Path planning for deformable linear objects". In: *IEEE Transactions on Robotics* 22.4 (2006), pp. 625–636.

[105] Jens Kober, J Andrew Bagnell, and Jan Peters. "Reinforcement learning in robotics: A survey". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1238–1274.

[106] Tuomas Haarnoja, Vitchyr Pong, Aurick Zhou, Murtaza Dalal, Pieter Abbeel, and Sergey Levine. "Composable deep reinforcement learning for robotic manipulation". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 6244–6251.

[107] Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. "End-to-end robotic reinforcement learning without reward engineering". In: *arXiv preprint arXiv:1904.07854* (2019).

[108] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. "End-to-end training of deep visuomotor policies". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1334–1373.

[109] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. "Multi-goal reinforcement learning: Challenging robotics environments and request for research". In: *arXiv preprint arXiv:1802.09464* (2018).

[110] Jan Matas, Stephen James, and Andrew J Davison. "Sim-to-real reinforcement learning for deformable object manipulation". In: *arXiv preprint arXiv:1806.07851* (2018).

[111] Gianluca Palli, Salvatore Pirozzi, Maurizio Indovini, Daniele De Gregorio, Riccardo Zanella, and Claudio Melchiorri. "Automatized Switchgear Wiring: An Outline of the WIRES Experiment Results". In: *Advances in Robotics Research: From Lab to Market*. Springer, 2020, pp. 107–123.

[112] Abdeslam Boularias, James Andrew Bagnell, and Anthony Stentz. "Learning to manipulate unknown objects in clutter by reinforcement". In: *Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015.

[113] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 4238–4245.

[114] Yuji Yamakawa, Akio Namiki, and Masatoshi Ishikawa. "Motion planning for dynamic knotting of a flexible rope with a high-speed robot arm". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2010, pp. 49–54.

[115] Hermann Mayer, Faustino Gomez, Daan Wierstra, Istvan Nagy, Alois Knoll, and Jürgen Schmidhuber. "A system for robotic heart surgery that learns to tie knots using recurrent neural networks". In: *Advanced Robotics* 22.13-14 (2008), pp. 1521–1537.

[116] Alex X Lee, Sandy H Huang, Dylan Hadfield-Menell, Eric Tzeng, and Pieter Abbeel. "Unifying scene registration and trajectory optimization for learning from demonstrations with application to manipulation of deformable objects". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 4402–4407.

[117] Weifu Wang, Dmitry Berenson, and Devin Balkcom. "An online method for tight-tolerance insertion tasks for string and rope". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 2488–2495.

[118] Matthias Rambow, Thomas Schauß, Martin Buss, and Sandra Hirche. "Autonomous manipulation of deformable objects based on teleoperated demonstrations". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 2809–2814.

[119] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[120] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[121] Joseph Redmon and Ali Farhadi. "YOLO9000: better, faster, stronger". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.

[122] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[123] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. "Prioritized experience replay". In: *arXiv preprint arXiv:1511.05952* (2015).

[124] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. "Hindsight experience replay". In: *Advances in neural information processing systems*. 2017, pp. 5048–5058.

[125] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[126] Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. "Yale-CMU-Berkeley dataset for robotic manipulation research". In: *The International Journal of Robotics Research* 36.3 (2017), pp. 261–268.

[127] Zhi-Hua Zhou. "A brief introduction to weakly supervised learning". In: *National Science Review* 5.1 (2018), pp. 44–53.

[128] Riccardo Zanella, Alessio Caporali, Kalyan Tadaka, Daniele De Gregorio, and Gianluca Palli. "Auto-generated Wires Dataset for Semantic Segmentation with Domain-Independence". In: *2021 International Conference on Computer, Control and Robotics (ICCCR)*. IEEE. 2021, pp. 292–298.

[129] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. ""GrabCut" interactive foreground extraction using iterated graph cuts". In: *ACM transactions on graphics (TOG)* 23.3 (2004), pp. 309–314.

[130] Hongyuan Zhu, Fanman Meng, Jianfei Cai, and Shijian Lu. "Beyond pixels: A comprehensive survey from bottom-up to semantic image segmentation and cosegmentation". In: *Journal of Visual Communication and Image Representation* 34 (2016), pp. 12–27.

[131] Hiba Ramadan, Chaymae Lachqar, and Hamid Tairi. "A survey of recent interactive image segmentation methods". In: *Computational Visual Media* (2020), pp. 1–30.

[132] Zhoumin Lu, Haiping Xu, and Genggeng Liu. "A survey of object co-segmentation". In: *IEEE Access* 7 (2019), pp. 62875–62893.

[133] Zilong Huang, Xinggang Wang, Jiasi Wang, Wenyu Liu, and Jingdong Wang. "Weakly-supervised semantic segmentation network with deep seeded region growing". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7014–7023.

[134] Daniele De Gregorio, Alessio Tonioni, Gianluca Palli, and Luigi Di Stefano. "Semiautomatic Labeling for Deep Learning in Robotics". In: *IEEE Transactions on Automation Science and Engineering* 17.2 (2019), pp. 611–620.

[135] Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker, Alberto Rodriguez, and Jianxiong Xiao. "Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge". In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2017, pp. 1386–1383.

[136] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. "Understanding real world indoor scenes with synthetic data". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4077–4085.

[137] Jonathan C Balloch, Varun Agrawal, Irfan Essa, and Sonia Chernova. "Unbiasing semantic segmentation for robot perception using synthetic data feature transfer". In: *arXiv preprint arXiv:1809.03676* (2018).

[138] Jonathan Tremblay, Thang To, and Stan Birchfield. "Falling things: A synthetic dataset for 3d object detection and pose estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 2038–2041.

[139] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. "Deep object pose estimation for semantic robotic grasping of household objects". In: *arXiv preprint arXiv:1809.10790* (2018).

[140] Baochen Sun, Jiashi Feng, and Kate Saenko. "Return of frustratingly easy domain adaptation". In: *arXiv preprint arXiv:1511.05547* (2015).

[141] Swami Sankaranarayanan, Yogesh Balaji, Arpit Jain, Ser Nam Lim, and Rama Chellappa. "Learning from synthetic data: Addressing domain shift for semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3752–3761.

[142] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. "Playing for data: Ground truth from computer games". In: *European conference on computer vision*. Springer. 2016, pp. 102–118.

[143] Joao Borrego, Atabak Dehban, Rui Figueiredo, Plinio Moreno, Alexandre Bernardino, and José Santos-Victor. "Applying domain randomization to synthetic data for object category detection". In: *arXiv preprint arXiv:1807.09834* (2018).

[144] Carl Doersch and Andrew Zisserman. "Sim2real transfer learning for 3D human pose estimation: motion to the rescue". In: *Advances in Neural Information Processing Systems*. 2019, pp. 12949–12961.

[145] Yiheng Zhang, Zhaofan Qiu, Ting Yao, Dong Liu, and Tao Mei. "Fully convolutional adaptation networks for semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6810–6818.

[146] Jihong Zhu, Benjamin Navarro, Philippe Fraisse, André Crosnier, and Andrea Cherubini. "Dual-arm robotic manipulation of flexible cables". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 479–484.

[147] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. "Encoder-decoder with atrous separable convolution for semantic image segmentation". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 801–818.

[148] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. "Deep high-resolution representation learning for visual recognition". In: *IEEE transactions on pattern analysis and machine intelligence* (2020).

[149] Benjamin Irving. "maskSLIC: regional superpixel generation with application to local pathology characterisation in medical images". In: *arXiv preprint arXiv:1606.09518* (2016).