electronics

MDPI

Article

Selective Grasping for Complex-Shaped Parts Using Topological Skeleton Extraction

Andrea Pennisi [1], Monica Sileo [2], Domenico Daniele Bloisi [1] and Francesco Pierri [2,*]

[1] Department of International Humanities and Social Sciences, UNINT International University of Rome, 00147 Rome, Italy; andrea.pennisi@gmail.com (A.P.); domenico.bloisi@unint.eu (D.D.B.)
[2] School of Engineering, University of Basilicata, 85100 Potenza, Italy; monica.sileo@unibas.it
* Correspondence: francesco.pierri@unibas.it; Tel.: +39-0971-205020

Abstract: To enhance the autonomy and flexibility of robotic systems, a crucial role is played by the capacity to perceive and grasp objects. More in detail, robot manipulators must detect the presence of the objects within their workspace, identify the grasping point, and compute a trajectory for approaching the objects with a pose of the end-effector suitable for performing the task. These can be challenging tasks in the presence of complex geometries, where multiple grasping-point candidates can be detected. In this paper, we present a novel approach for dealing with complex-shaped automotive parts consisting of a deep-learning-based method for topological skeleton extraction and an active grasping pose selection mechanism. In particular, we use a modified version of the well-known Lightweight OpenPose algorithm to estimate the topological skeleton of real-world automotive parts. The estimated skeleton is used to select the best grasping pose for the object at hand. Our approach is designed to be more computationally efficient with respect to other existing grasping pose detection methods. Quantitative experiments conducted with a 7 DoF manipulator on different real-world automotive components demonstrate the effectiveness of the proposed approach with a success rate of 87.04%.

Keywords: computer vision for manufacturing; deep learning in grasping and manipulation; visual learning

Citation: Pennisi, A.; Sileo, M.; Bloisi, D.D.; Pierri, F. Selective Grasping for Complex-Shaped Parts Using Topological Skeleton Extraction. *Electronics* **2024**, *13*, 3021. https://doi.org/10.3390/electronics13153021

Academic Editors: Krzysztof Okarma and Piotr Lech

Received: 4 June 2024
Revised: 23 July 2024
Accepted: 29 July 2024
Published: 31 July 2024

1. Introduction

Components with complex geometrical shapes are largely used in the manufacturing industry, e.g., in the automotive sector. Using robots to handle complex-shaped parts is still a challenging task due to perception, planning, and reasoning problems. In particular, uncertainties in the position of the object to grasp and perception noise due to reflective materials are common challenges in industrial scenarios.

Grasping objects with complex geometries can be roughly classified into model-based methods that rely on pre-existing 3D models and learning-based techniques that employ machine learning to predict grasping points. Both the approaches need to perceive the external environment using vision-based algorithms, based on cameras and point clouds for object detection, segmentation, and pose estimation, as in [1,2], or tactile-based strategies, as in [3], requiring sensors for force measurement, haptic feedback, and slip detection. Hybrid approaches combine multiple methods for robustness, including multi-sensor fusion and active perception.

In this paper, we present a complete pipeline for handling complex-shaped automotive parts using a 7 DoF robot manipulator. In particular, we adopt a deep learning-based approach to design a multi-object detector aimed to extract the topological skeleton belonging to the part to grasp to precisely estimate its pose. After estimating the pose, a selection of the best grasping pose is carried out to increase the chance of grasping the object successfully.

The contribution of this work is three-fold.

1. The skeleton extraction process provides a representation of the object pose in 3D space. Therefore, it also allows a precise estimation of the orientation of the object.
2. We use MobileNetV3 to replace the original MobileNetV1 backbone in the skeleton extraction network, and we customize it to detect the skeleton of industrial objects with complex geometry from both front and back views, even if the objects have different shapes on either side.
3. The grasping pose selection is carried out using a dynamic approach. This means that an error in the skeleton extraction is autonomously detected and the robot actively modifies its position to better perform the grasping.

We have conducted several experiments with real-world automotive parts to validate our approach, which can detect the object's keypoints from upside-down views, without any constraint.

The remainder of the paper is organized as follows. Section 2 contains a brief description of existing related work. Our method is presented in Section 3. Experiments demonstrating the effectiveness of the proposed approach are shown in Section 4. Finally, conclusions and future directions are drawn in Section 5.

2. Related Work

In the last few years, thanks to the availability of powerful GPUs, deep learning methods have become suitable for dealing with grasping-point detection. They have proven capable of replacing traditional analytical approaches based on geometrical properties, physics models, and force analytics. A Convolutional Neural Network (CNN) architecture named GraspNet, able to segment graspable regions on the surfaces of objects, has been presented in [4]. In [5], a CNN, in combination with the information provided by a depth camera, has been used to detect the presence of the object and the best grasping pose. Several approaches were proposed to improve the accuracy of deep CNN, see, e.g., [6,7], but they usually require long computation time (i.e., of the order of seconds).

More efficient approaches, requiring only depth images, have been proposed in [8,9]. More in detail, in [8], a Deep Convolutional Neural Network has been trained in a simulated environment to learn grasping-relevant features and return a single-grasp solution for each object. In [9], the so-called generative grasping convolutional neural network (GG-CNN) has been proposed. It allows direct evaluation of the grasp quality and pose of grasps for every pixel in an input depth image, and it is fast enough to perform grasping in dynamic environments. The GG-CNN performance has been improved by introducing the GG-CNN2 [10], which is a CNN based on the semantic segmentation architecture of [11]. A common characteristic of deep-learning-based methods for grasping-point detection is the need to calculate the grasping quality value for each pixel in the image at hand, which is extremely time-consuming.

When the object knowledge and the grasp pose candidates are not available, it is possible to approximate the object using shape primitives, e.g., using multiview measurements [12] or identifying features in sensory data [13]. The method proposed in [14] consists of selecting grasp pose candidates after locating areas where a successful grasp had already been experienced. In [1], grasping partially known objects in unstructured environments is proposed based on an extension to the industrial context of the well-known technique of Background Subtraction [15]. Thanks to the spreading of low-cost depth sensors, many 3D registration algorithms have been exploited to handle the object grasping problem. For example, in [2], a model of the object to be grasped is generated using a set of point clouds acquired from different positions, and the nominal grasping pose is fixed. Subsequently, this model is compared with the runtime object view to compute the current grasping pose.

In this work, we propose a skeleton-based approach for detecting the grasping poses, which is inherently less computationally demanding due to the compact representation of the object via the skeleton.

A qualitative and quantitative comparison between our approach and the most relevant papers described in this section is shown in Table 1. This comparison takes into account not only the results but also the limitations of each work.

Table 1. Comparison table between different object grasping approaches.

Methods	Applications	Quantitative Results	Limitations
CNN architecture with a DDF module [4]	Real-time robotic grasping	90% of accuracy on Cornell grasp dataset	Error in predict orientation for some objects
Structure based on faster R-CNN and DACAB [5]	Object grasping with mobile manipulator	86.3% of success rate	Inefficient search method
GQ_CNN to classify robust grasping [6]	Grasping household objects	99% of precision	Long computational time
DCNN based on depth images to predict grasp pose [8]	Grasping of unknown objects	92% (70%) of precision with cylindrical-shaped (box-shaped) objects	Generation of a single-grasp solution for a single object
NN for learning prototypical parts [14]	Grasping of similar objects	N.A.	Grasping of complex-shaped objects with never-before-seen features.
Topological skeleton extraction (this work)	Grasping of complex-shaped automotive parts	87.04% of success rate	Needs a good camera calibration

3. Proposed Method

Figure 1 shows the overall functional architecture of our approach. It is made of four main modules, namely Visual Data Acquisition, Topological Skeleton Extraction, Grasping Pose Selection, and Robot Grasping. Each module is detailed below.

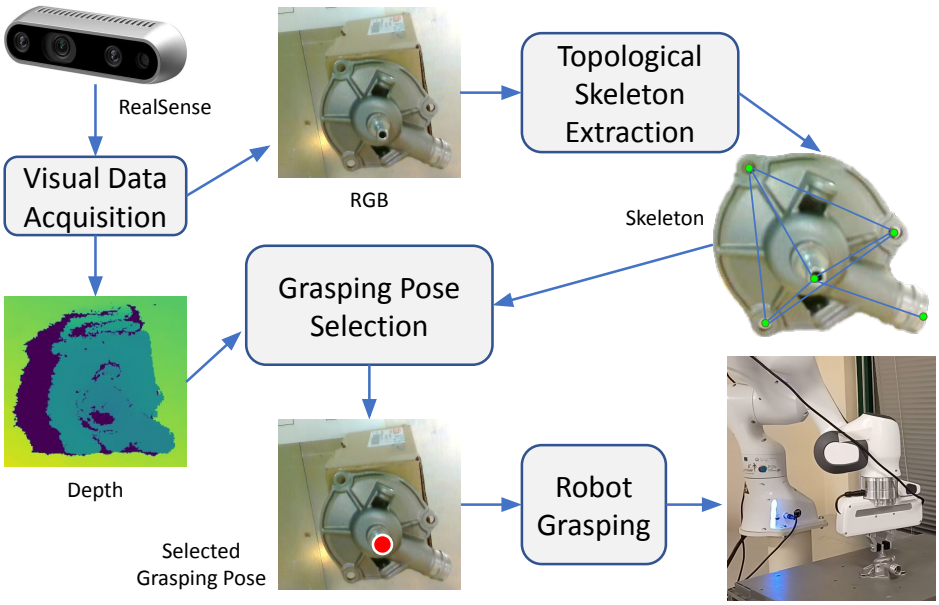

Figure 1. Functional architecture of the proposed approach.

3.1. Visual Data Acquisition

Visual data (both RGB and depth) are acquired using an Intel Realsense D435 RGBD camera, mounted on the end-effector via a 3D printed support in the so-called *eye-in-hand* configuration. Figure 2 shows the reference frames attached to the robot end-effector, $\mathcal{F}_e$, and the camera, $\mathcal{F}_c$.

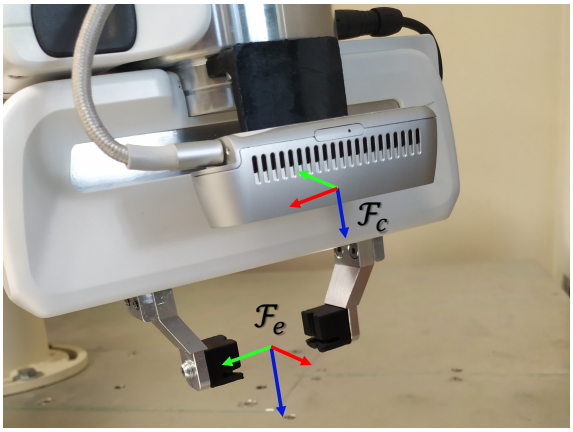

Figure 2. End-effector and camera reference frames.

The Intel Realsense D435 camera has a minimum depth distance beyond which it is not able to provide a depth measure approximately equal to 28 cm and the camera data acquisition requires the `realsense-ros` library since communication between the modules takes place through the Robot Operating System (ROS). The camera has been previously calibrated using 30 images of a 2D chessboard flat pattern. The calibration process includes both intrinsic and extrinsic calibrations. The first is aimed at determining the camera parameters that describe how the camera transforms the 3D coordinates of the scene into the 2D coordinates of the image, like the focal length, the principal point, and the optical distortions, while the second one provides the parameters which describe the rigid transformation that maps the 3D coordinates of the real world to the 3D coordinates of the camera's reference system. The calibration procedure implementation proposed by the VISP library [16], based on [17,18], has been adopted using a chessboard composed of 9×6 squares with dimensions of 0.02645 m.

It is worth remembering that a good calibration procedure is crucial for the success of the grasping procedure since it ensures an accurate perception of the environment, enabling precise identification and positioning of the points in three-dimensional space for successful manipulation.

3.2. Topological Skeleton Extraction

The proposed method has been developed for objects that:

- are rigid, as it is not applicable to deformable objects;
- are not perfectly symmetrical since although it is possible to define a non-symmetric topological skeleton, the detector may become confused during the extraction process due to symmetrical features.

In this work, we focus on real automotive parts, including two crankcase oil separator covers made of cast iron and plastic and an air pipe. The selected objects have an increasing level of difficulty. The first object, the cast iron crankcase oil separator cover, exhibits a high degree of symmetry with multiple grasping points and can be grasped by a cylindrical part, therefore reducing the impact of the robot orientation errors around the axis of the pin. The second object, the plastic crankcase oil separator cover, also exhibits a high degree of symmetry with various grasping points but must be grasped with a specific orientation. Finally, the air pipe has a complex shape, lacks symmetry, and has only two available grasping points, representing the most challenging task for the robot.

We decided to model their skeletons considering a few keypoints, some of which correspond to the potential grasping points for lifting that object with the robot manipulator (see the upper right part of Figure 1). To detect the Topological Skeleton (TS) of the objects to be grasped, we consider Lightweight OpenPose [19], whose architecture consists of three main components: a feature extractor, a TS estimator, and a Part Affinity Fields (PAF) network.

In comparison to the original OpenPose [20], we chose Lightweight OpenPose because the high computational demand of the former method makes it less applicable in real-time applications on devices with little processing power. OpenPose employs a two-branch, multi-stage CNN architecture. The first branch predicts part confidence maps (PCM) for body parts, and the second branch predicts part affinity fields (PAF) to model the connections between body parts. The architecture involves several stages of convolutions to refine these predictions iteratively, resulting in high accuracy at the cost of increased computational load. Light OpenPose, on the other hand, modifies the original architecture to reduce complexity and improve efficiency. The approach reduces the number of convolutional layers and stages and uses depthwise separable convolutions in place of standard convolutions to reduce the number of parameters and operations. Moreover, the backbone network uses MobileNet or ShuffleNet in place of the heavier VGG19 or ResNet used in the original OpenPose and optimizes the computation of part affinity fields to strike a balance between accuracy and efficiency.

Feature extraction. The original Lightweight OpenPose uses a MobileNetV1 network that is optimized for reaching real-time feature extraction. MobileNet is a family of neural network architectures designed for efficient deployment on mobile and embedded devices with limited computational resources. The key feature of MobileNet is its use of depthwise separable convolutions, which can significantly reduce the number of parameters and computations required while maintaining high accuracy.

While MobileNetV1 is a highly effective neural network, it does have some limitations and drawbacks that should be considered. For instance, it has limited accuracy because it is designed to balance model size and accuracy. It may not achieve the same level of accuracy as larger and more complex neural networks, especially on challenging objects where the keypoints (joints) are not evident. The depthwise separable convolution operation used in MobileNetV1 can be less expressive than traditional convolutional operations and may not be able to capture all the important features of an image.

For the above reasons, in this work, we propose to replace MobileNetV1 with MobileNetV3 [21] for the feature extraction step. MobileNetV3 has been designed to address the limitations of MobileNetV1 while maintaining efficiency. The architecture of the MobileNetV3 network used in this work is shown in Figure 3.

MobileNet V3 has two main variants: (1) *MobileNet V3-Large* designed for higher accuracy applications, with more layers and channels, and (2) *MobileNet V3-Small* optimized for resource-constrained environments, trading off some accuracy for reduced computational demand. Our choice fell on the latter one. MobileNet V3 introduces several new components, such as Inverted Residual Blocks to maintain a high degree of efficiency, Squeeze-and-Excitation (SE) Modules to improve the representational power of the model by recalibrating channel-wise feature responses and the H-Swish Activation Function.

The hard-swish function is a non-linear activation function that is designed to be more efficient than traditional activation functions such as ReLU. The hard-swish function is defined as

$$\text{h-swish}(x) = x\frac{ReLU6(x+3)}{6}\ , \tag{1}$$

where $ReLU6(x) = min(max(x, 0), 6)$ is a clipped ReLU function that outputs values between 0 and 6, still providing a non-linear behavior while increasing the computational speed with respect to the standard $ReLU$ function.

Another important feature of MobileNetV3 is the use of a squeeze-and-excitation (SE) module. The SE module is a simple and efficient way to improve the representational power

of the network (i.e., the ability to learn and represent complex patterns and features in the input data). It works by learning channel-wise scaling factors that are used to selectively enhance informative features in the network. The SE module is added to each bottleneck block in the MobileNetV3 architecture, contributing to increasing the accuracy with respect to MobileNetV1.

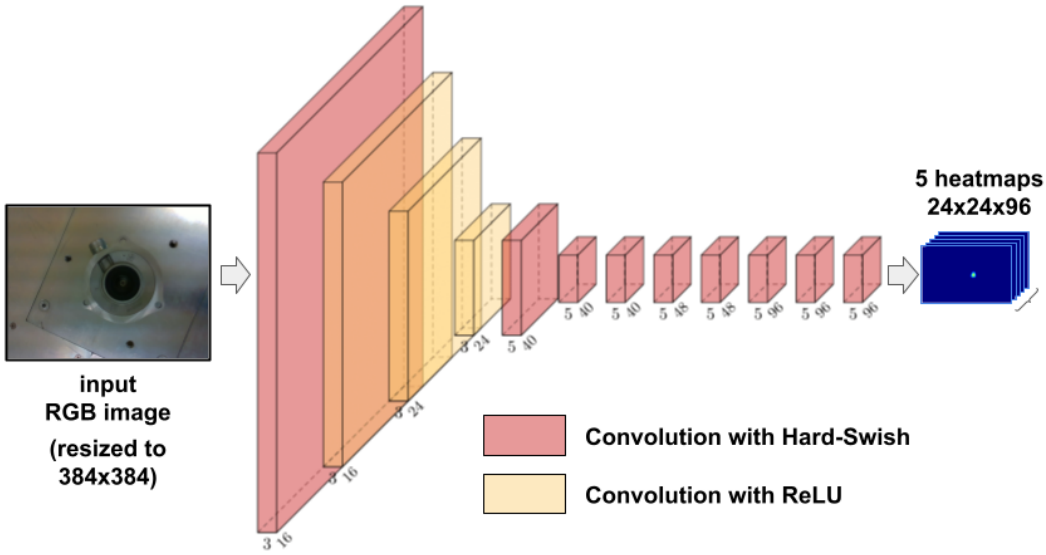

Figure 3. MobileNetV3 architecture.

MobileNetV3 also introduces a new technique called the mobile inverted bottleneck convolution (MBConv), which is a modified form of the depthwise separable convolution used in MobileNetV1. The MBConv block consists of three types of convolutions: a 1×1 convolution to expand the number of channels, a depthwise convolution to perform spatial filtering, and a 1×1 convolution to reduce the number of channels back to the original size. The MBConv block also includes a shortcut connection that allows the gradient to flow directly from the input to the output. MBConv block helps in increasing the expressiveness of the model with respect to MobileNetV1.

Finally, MobileNetV3 includes a middle-flow block that is used to maintain a high level of accuracy while minimizing the number of computations required. It also uses a dynamic convolution operation that adapts to the input data. The details of the parameters used for each block are described in Table 2. The input image is resized to 384×384, and the output is a set of $24 \times 24 \times 96$ feature maps, one for each keypoint and one for the background.

Table 2. The MobileNetV3 network architecture used in this paper. *HS* = hard-swish, *RE* = ReLU, *s* = stride.

Input	Operator	Exp Size	*#out*	SE	NL	s
$384^2 \times 3$	conv2d, 3×3	-	16	-	HS	2
$192^2 \times 16$	bneck, 3×3	16	16	x	RE	2
$96^2 \times 16$	bneck, 3×3	72	24	-	RE	2
$48^2 \times 24$	bneck, 3×3	88	24	-	RE	1
$48^2 \times 24$	bneck, 5×5	96	40	x	HS	2
$24^2 \times 40$	bneck, 5×5	240	40	x	HS	1
$24^2 \times 40$	bneck, 5×5	240	40	x	HS	1
$24^2 \times 40$	bneck, 5×5	120	48	x	HS	1
$24^2 \times 48$	bneck, 5×5	144	48	x	HS	1
$24^2 \times 48$	bneck, 5×5	288	96	x	HS	1
$24^2 \times 96$	bneck, 5×5	576	96	x	HS	1
$24^2 \times 96$	bneck, 5×5	576	96	x	HS	1

TS estimation. The feature maps from MobileNetV3 are the input to generate a set of candidate key points for each object part in the image. In fact, the feature maps capture the spatial information in the input image and provide a rich representation of the image that can be used to detect keypoints. MobileNetV3 adds a custom head to predict keypoint locations, which consists of o a series of convolutional layers that generate heatmaps, refining the features extracted by the backbone and generating heatmaps for each keypoint. Figure 4 shows an example of the TS estimator output for the cast iron crankcase oil separator cover, which consists of five heatmaps, one for each considered keypoint. Each heatmap has the same spatial resolution as the feature maps and is normalized to have values between 0 and 1. Each pixel in the heatmap indicates the likelihood that the corresponding body part is present at that location in the image.

PAF network. It takes the feature maps generated by the feature extractor as input and outputs a set of PAF feature maps, one for each pair of the detected keypoints. The PAF feature maps encode the direction and strength of the connections between keypoints using a two-channel representation, where each channel encodes a different aspect of the connection. Specifically, one channel encodes the unit vector that represents the direction of the connection, while the other channel encodes the confidence score that represents the strength of the connection.

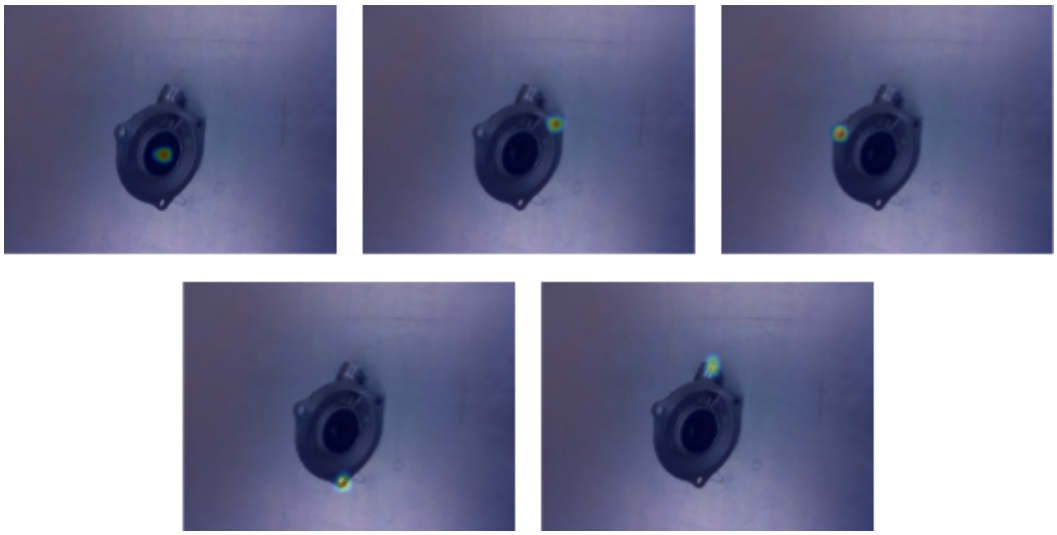

Figure 4. Heatmap examples for cast iron crankcase oil separator cover. There are five heatmaps corresponding to the considered keypoints.

Final TS computation. Once the PAF and heatmaps are generated, they are used together to group the individual keypoints into the final TS. The final TS is obtained by first identifying the candidate connections using the PAFs and then scoring the connections based on the likelihood that they form a valid connection. The connections are then used to construct the final TS by connecting the individual keypoints into a complete object TS.

Once the keypoints are calculated, we use the depth information for building the final 3D TS given the set of keypoints from Lightweight OpenPose. Figure 5 shows some examples of final TSs for the three objects considered, where it is possible to note the robustness of the proposed TS extraction approach with respect to different views of the object, to photochromic changes and partial occlusions. The approach is also working with multiple instances of the object.

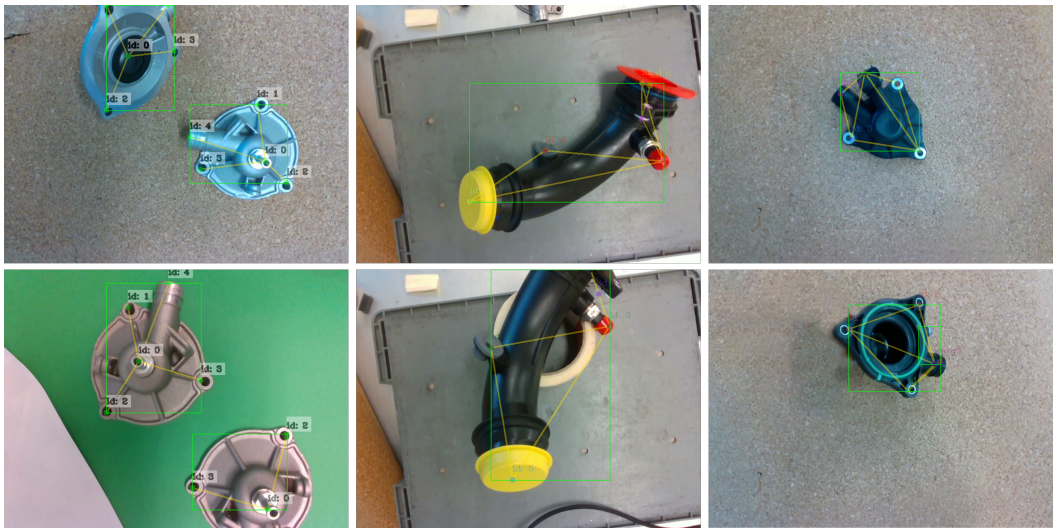

Figure 5. TS extraction examples on different objects: the cast iron crankcase oil separator cover on the left, the air pipe in the middle, and the plastic crankcase oil separator cover on the right. Our approach is robust to different views of the same object, to photochromic changes, and partial occlusions.

3.3. Grasping Pose Selection

After the selection of the N_k keypoints for the TS extraction, these keypoints are also identified within the CAD model of the object through 3D modeling software. This results in the generation of a nominal three-dimensional representation of the TS, TS_N, in the CAD coordinate system, $\mathcal{F}_f$. Moreover, the poses in $\mathcal{F}_f$ of all possible N_g grasping reference frames (see Figure 6), expressed via the (4×4) homogeneous transformation matrix [22], $\boldsymbol{T}^f_{g_j}$, $j = 1, \ldots, N_g$, can be localized on the model. For the sake of clarity, let us assume that each grasping point coincides with a keypoint.

Figure 6. The grasping frames for the considered objects: cast iron crankcase oil separator cover (**top row)**, plastic crankcase oil separator cover (**middle row**), and air pipe (**bottom row**).

Then, given all possible combinations of three keypoints

$$\mathcal{S}_t = \left\{ t_i,\ i = 1, \ldots, N_t = \binom{N_k}{3} : t_i = (P_j, P_l, P_m),\ j, l, m \in \{1, \ldots, N_k\},\ j \neq l \neq m \right\},$$

for each triple t_i, a plane is identified via a coordinate frame attached to it, whose pose is denoted by the homogeneous transformation matrix $\boldsymbol{T}^f_{t_i}$. For each grasping reference frame, the relative pose with respect to the ith plane can be determined as

$$\boldsymbol{T}^{t_i}_{g_j} = \left(\boldsymbol{T}^f_{t_i}\right)^{-1} \boldsymbol{T}^f_{g_j}\ . \tag{2}$$

Thus, for each grasping point, a list of N_t transformation matrices, $\boldsymbol{T}^{t_i}_{g_j}$, representing the grasping frame poses in the plane frame, can be computed. This set of operations, summarized in Algorithm 1, is performed only once.

Algorithm 1: Pre-processing algorithm

Input : TS_N, N_g, N_t, $\boldsymbol{T}^f_{g_j}$ $(j = 1, \ldots, N_g)$
Output: $\boldsymbol{T}^{t_i}_{g_j}$ $(i = 1, \ldots, N_t; j = 1, \ldots, N_g)$

```
for each triple t_i of keypoints in TS_N do
    Compute T^f_{t_i}
end for
for i = 1, ..., N_t do
    for j = 1, ..., N_g do
        T^{t_i}_{g_j} = (T^f_{t_i})^{-1} T^f_{g_j}
    end for
end for
return T^{t_i}_{g_j}
```

At runtime, the following steps are executed:

1. A YOLO detector [23] is adopted to distinguish between the objects. YOLO has been chosen since it is faster than classifier-based systems but with similar accuracy and makes predictions with a single network evaluation by considering object detection as a single regression problem, and this leads to high accuracy performance. Moreover, YOLO can detect and classify multiple objects simultaneously within an image.
2. The current 3D TS, TS_C, is extracted.
3. The grasping point closest to the camera, $\boldsymbol{p}^c_{cc}$, is selected as the best one.

4a. If at least 3 keypoints are visible, a set of three keypoints, t_k, is used to compute the corresponding plane in the camera frame, $\boldsymbol{T}^c_{t_k}$, and to select the homogeneous transformation matrix, $\boldsymbol{T}^{t_k}_{g_{cc}}$, that identifies the grasping pose in the plane frame. Then, the procedure continues with the step 5.

4b. If only 2 or fewer keypoints are visible, the robot starts moving in a circle around the center of the object bounding box to acquire a new image from a different point of view. Then, the procedure comes back to the step 1.

5. The grasping pose in camera frame is computed as

$$\boldsymbol{T}^c_{g_{cc}} = \boldsymbol{T}^c_{t_k} \boldsymbol{T}^{t_k}_{g_{cc}}\ .$$

This procedure is summarized in Algorithm 2.

Let us define the homogeneous transformation matrix $\boldsymbol{T}^e_c$, i.e., the constant homogeneous matrix performing the transformation between the camera frame and the end-effector frame, obtained via the calibration method described in Section 3.1.

Algorithm 2: Runtime algorithm

Input : ${}^{obj}\boldsymbol{T}_{g_j}^{t_i}$ for each object
Output: $\boldsymbol{T}_{g_{cc}}^{c}$

Get image from camera
Detect current object from image
$TS_C \leftarrow$ extract TS
$\mathcal{T} = \boldsymbol{T}_{g_j}^{t_i}\Big|_{i=1,\ldots,N_t;\ j=1,\ldots,N_g} \leftarrow$ get the list of transformations relative to the current object
$\boldsymbol{p}_{cc}^{c} \leftarrow$ extract the grasping point closest to the camera
if *visible_keypoints* ≥ 3 **then**
    $t_k \leftarrow$ extract a triple of visible keypoints in TS_C
    Compute $\boldsymbol{T}_{t_k}^{c}$
    Extract $\boldsymbol{T}_{g_{cc}}^{t_k}$ from $\mathcal{T}$
else
    move the robot to a different point of view
    GO TO 1
end if
$\boldsymbol{T}_{g_{cc}}^{c} \leftarrow \boldsymbol{T}_{t_k}^{c}\,\boldsymbol{T}_{g_{cc}}^{t_k}$
return $\boldsymbol{T}_{g_{cc}}^{c}$

To capture the grasping pose in the inertial frame, $\boldsymbol{T}_{g_{cc}}^{c}$ is transformed as follows

$$\boldsymbol{T}_{g_{cc}} = \boldsymbol{T}_{e}\,\boldsymbol{T}_{c}^{e}\,\boldsymbol{T}_{g_{cc}}^{c}\ , \tag{3}$$

where $\boldsymbol{T}_e$ is the homogeneous matrix representing the pose of the end-effector in the inertial frame.

Remark 1. *It is worth noting that if the grasping point is not coincident with a keypoint, the above procedure is still applicable, but a further constant transformation needs to be applied to link the grasping point to one of the keypoints belonging to the plane.*

3.4. Robot Grasping

To perform the grasp, the end-effector must be commanded to align its reference frame to the grasping reference frame. The trajectory is planned by assigning a sequence of three points: the first one is the view pose of the robot, the intermediate one is the *approach* point, i.e., a point positioned along the z axis of the grasping reference frame at a distance of 10 cm to the origin, and the last one is the estimated grasping position, $\hat{\boldsymbol{p}}_g$. More in detail, the end-effector desired position, $\boldsymbol{p}_{e,d}(t)$, is defined as

$$\boldsymbol{p}_{e,d}(t) = \begin{cases} \boldsymbol{p}_0 + \dfrac{s_1(t)}{\|\boldsymbol{p}_a - \boldsymbol{p}_0\|}(\boldsymbol{p}_a - \boldsymbol{p}_0) & \text{for } 0 \leq t \leq t_a \\ \\ \boldsymbol{p}_a + \dfrac{s_2(t)}{\|\hat{\boldsymbol{p}}_g - \boldsymbol{p}_a\|}(\hat{\boldsymbol{p}}_g - \boldsymbol{p}_a) & \text{for } t_a < t \leq t_f \end{cases} . \tag{4}$$

where $\boldsymbol{p}_0$ is the view position and $\boldsymbol{p}_a$ is the approach point position, $s_1(t)$ ($s_2(t)$) is the *arc length* form $\boldsymbol{p}_0$ to $\boldsymbol{p}_a$ (from $\boldsymbol{p}_a$ to $\hat{\boldsymbol{p}}_g$). To ensure continuous acceleration and velocities at the path points, both for $s_1(t)$ and $s_2(t)$, the time-law can be designed as a quintic polynomial. Regarding the time instants, t_f is the duration of the motion, and t_a is the intermediate time instant at the approach point that is chosen to have a fast motion until the approach point and a slow motion in the object's proximity.

Regarding the end-effector orientation, it is planned to reach the same orientation of the estimated grasping pose, $\widehat{\boldsymbol{R}}_g$, at the approach point and to keep such orientation constant during the last part of the path.

The planned trajectory in terms of position and orientation is the input of the closed-loop inverse kinematics algorithm [22] aimed at computing the reference values of the joint positions and velocities. Let denote with $\boldsymbol{p}_e(t)$ and $\boldsymbol{R}_e(t)$ the end-effector position and orientation, respectively, and with $\boldsymbol{R}_{e,d}(t)$ the end-effector desired orientation. The robot joint velocity references, $\dot{\boldsymbol{q}}_r(t)$, are computed as

$$\dot{\boldsymbol{q}}_r(t) = \boldsymbol{J}^{\dagger}(\boldsymbol{q}(t))(\dot{\boldsymbol{v}}_{e,d}(t) + \boldsymbol{K}\boldsymbol{e}(t)), \tag{5}$$

where $\boldsymbol{J}^{\dagger}(\boldsymbol{q}(t))$ denotes the right pseudo-inverse of the robot Jacobian matrix, $\boldsymbol{K} \in \mathbb{R}^{6\times 6}$ is a positive definite matrix gain, $\boldsymbol{v}_{e,d} = \begin{bmatrix} \dot{\boldsymbol{p}}_{e,d}^{\mathrm{T}} & \boldsymbol{\omega}_{e,d}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ is the desired end-effector linear and angular velocity, and $\boldsymbol{e}$ is the tracking error defined as

$$\boldsymbol{e} = \begin{bmatrix} \boldsymbol{p}_{e,d} - \boldsymbol{p}_e \\ \eta_e\boldsymbol{\epsilon}_{e,d} - \eta_{e,d}\boldsymbol{\epsilon}_e - \boldsymbol{S}(\boldsymbol{\epsilon}_{e,d})\boldsymbol{\epsilon}_e \end{bmatrix}, \tag{6}$$

where $\mathcal{Q}_e = \{\eta_e, \boldsymbol{\epsilon}_e\}$ and $\mathcal{Q}_{e,d} = \{\eta_{e,d}, \boldsymbol{\epsilon}_{e,d}\}$ are the unit quaternion extracted from $\boldsymbol{R}_e$ and $\boldsymbol{R}_{e,d}$, respectively, and $\boldsymbol{S}(\cdot)$ is the skew-symmetric matrix operator performing the cross product [22].

A flowchart representation highlighting the whole process is given in Figure 7.

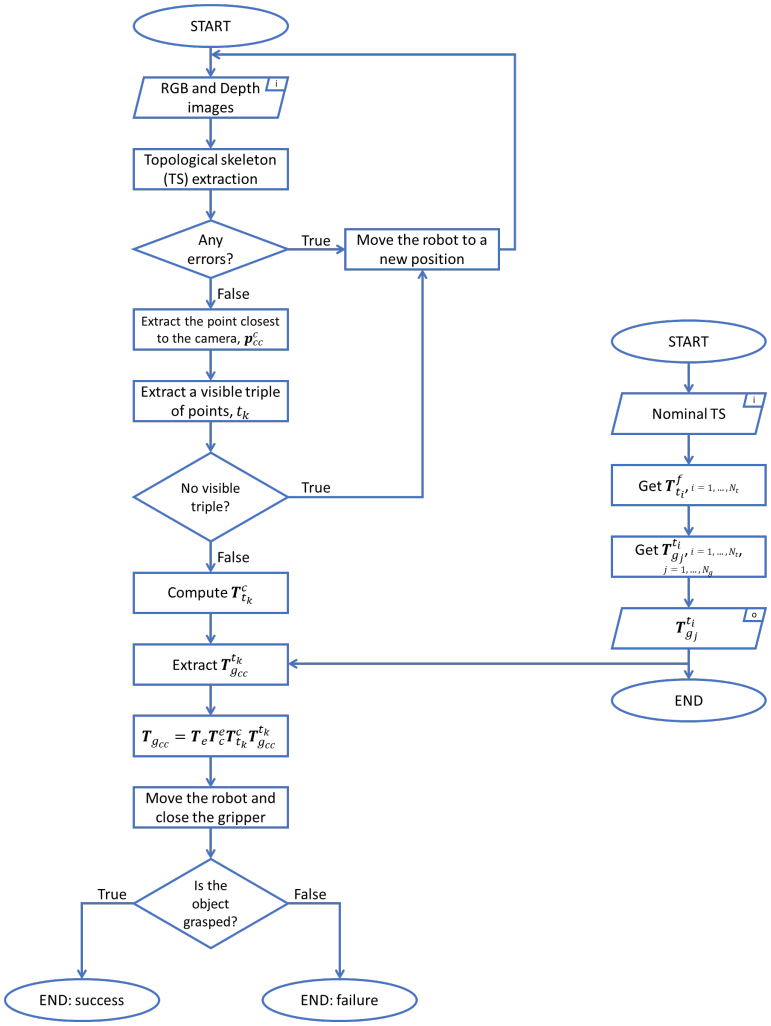

Figure 7. Flowchart representation of the whole process.

4. Experimental Results

The experimental setup consists of an Intel RealSense D435 camera mounted on a Franka Emika Panda robot manipulator, characterized by 7 revolute joints. The robot can be controlled by means of the Franka Control Interface (FCI) and the `libfranka` C++ open-source library, which directly controls the robot with an external workstation through an ethernet connection. In this work, the `franka_ros` meta-package, which integrates `libfranka` into ROS, has been used. The workstation runs Ubuntu 18.04 LTS and a real-time kernel on an Intel Xeon 3.7 GHz CPU with 32 GB RAM. We have conducted experiments with the three considered objects shown in Figure 6, and the quantitative results are reported below.

4.1. TS Extraction Results

Using Coco Annotator [24], 5992 images have been annotated. The labeled data have been split into Training, Validation, and Test sets composed of 4618, 229, and 1145 images, respectively. Table 3 shows the number of images in the Training, Validation, and Test sets for each considered object.

Table 3. Number of sample images used in Training, Validation, and Test sets for the considered objects.

Object	Training	Validation	Test
Cast iron crankcase oil separator cover	1440	60	300
Air pipe	1406	46	228
Plastic crankcase oil separator cover	1772	123	617

The metric we used for evaluating the TS detection is the Object Keypoint Similarity (*OKS*) [25], defined as follows:

$$OKS = \frac{\sum_{i\in[0,N-1]} exp\left(\frac{-d_i^2}{2s^2k_i^2}\right)\delta(v_i > 0)}{\sum_{i\in[0,N-1]} \delta(v_i > 0)} \tag{7}$$

where:

- s is the object scale;
- d_i is the distance of the predicted keypoint i from the ground truth;
- k_i is a per-keypoint constant that controls the falloff;
- v_i is the visibility flag.

OKS is calculated for each sample representing an object. The visibility flag takes into account if a point is visible or not: if the keypoint is labeled, $\delta(v_i > 0)$ is 1, else it is 0 without considering occluded keypoints.

In our scenario, we used *OKS* to compute the True Positive (TP), False Negative (FN), False Positive (FP), and True Negative (TN) detections. If a detection has $OKS > threshold$, it is considered to be a TP; otherwise, as an FP. In particular, we considered two thresholds, namely 0.5 and 0.75, and calculated the following metrics: Precision, Recall, F1-score, and Average Precision (AP). Table 4 shows the results of our algorithm for a test set of 1145 images.

Table 4. Results of the TS detector at different thresholds for a test set of 1145 images.

Threshold	Precision	Recall	F1-Score	AP
0.5	0.92	0.90	0.91	0.82
0.75	0.86	0.89	0.87	0.72

To compute the runtime performance of our TS extractor, we tested it on a subset of 60 images using an NVIDIA RTX A5500, obtaining an average execution time of 0.012 s and

a standard deviation of 0.0018 s. On a subset of 40 images, using an NVIDIA QUADRO T2000, the average execution time is 0.019 s, and the standard deviation is 0.0025 s.

4.2. Object Detector Results

For training the object detector, we annotated 750 images of size 640 × 480 using the LabelImg annotation tool [26]. We split the dataset into Train, Validation, and Test sets composed of 450, 150, and 150 images, respectively. After the training stage, the mean average precision on the test set is 97.32%, and the success rate is 96.70%. The inference on the images has been executed on an NVIDIA QUADRO T2000. On a subset of 40 images, the average execution time is 0.323 s, while the standard deviation is 0.0615 s.

4.3. Robot Grasping Results

Let us define the estimation grasping position and orientation errors as

$$\boldsymbol{e}_p^e = \boldsymbol{p}_g^e - \hat{\boldsymbol{p}}_g^e, \tag{8}$$

$$\boldsymbol{e}_\phi^e = \boldsymbol{\phi}_g^e - \hat{\boldsymbol{\phi}}_g^e, \tag{9}$$

where $\boldsymbol{p}_g^e$ is the actual grasping position while $\hat{\boldsymbol{p}}_g^e$ is the estimate provided by the visual algorithm. Regarding the orientation, $\boldsymbol{\phi}_g^e$ ($\hat{\boldsymbol{\phi}}_g^e$) is the Euler angles extracted from the actual (estimated) grasping pose. The adoption of Euler angles in lieu of quaternions as in (6) provides a clearer physical interpretation of the orientation errors. The superscript e denotes that the variables are expressed in the end-effector frame (see Figure 2).

To have statistically significant results, 54 grasping tests (20 for the cast iron crankcase oil separator cover, 19 for the air pipe, and 15 for the plastic crankcase oil separator cover) have been conducted by placing the objects in different configurations, different light conditions, and with different backgrounds in a way to let the robot explore all the possible grasping poses. A grasping test is considered successful if the gripper holds the object with a stable grasping for 10 s. A set of snapshots of the grasping procedure is shown in Figure 8, where the top row refers to a successful test and the bottom row refers to a failure.

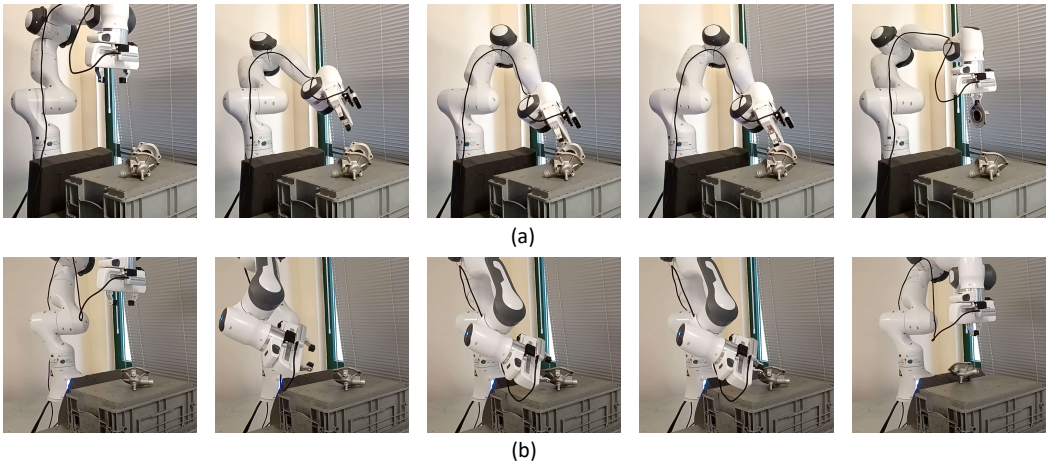

Figure 8. Snapshots of two grasping cases. (**a**) Successful grasp. (**b**) Failure.

Only 7 experiments (2 for the cast iron crankcase oil separator cover, 3 for the air pipe, and 2 for the plastic crankcase oil separator cover) experienced a failure. Thus, a success rate of 87.04% has been obtained. Tables 5–7 show the mean position and orientation errors and the corresponding standard deviation for the successful tests.

Table 5. Mean errors for the cast iron crankcase oil separator cover.

Successful Tests	$e^e_{p_x}$ [mm]	$e^e_{p_y}$ [mm]	$e^e_{\phi_x}$ [deg]	$e^e_{\phi_y}$ [deg]
1	0.734	9.471	3.128	3.515
2	6.867	0.717	3.284	3.091
3	1.081	5.857	6.092	12.433
4	13.032	10.752	3.132	6.256
5	1.796	0.775	6.801	0.77
6	3.832	0.759	3.913	3.546
7	3.629	1.531	2.981	3.416
8	1.324	6.747	0.674	0.92
9	3.073	7.437	0.62	35.308
10	1.293	1.021	0.952	1.605
11	3.916	1.631	0.391	6.217
12	4.62	2.697	3.044	4.508
13	2.368	4.09	9.12	3.953
14	0.463	6.314	2.456	10.075
15	1.816	2.214	2.217	5.806
16	2.958	4.751	1.842	7.763
17	2.86	8.736	1.354	4.138
18	3.018	0.747	13.855	2.16
Mean	3.26	4.236	3.659	6.415
Standard deviation	2.820	3.278	3.329	7.607

Table 6. Mean errors for the air pipe.

Successful Tests	$e^e_{p_x}$ [mm]	$e^e_{p_y}$ [mm]	$e^e_{\phi_x}$ [deg]	$e^e_{\phi_y}$ [deg]
1	0.043	0.989	2.439	8.109
2	9.586	17.223	4.597	0.555
3	1.098	2.612	16.723	13.204
4	3.777	6.188	19.545	10.702
5	7.213	0.227	3.663	1.042
6	3.882	1.516	1.684	0.315
7	2.897	2.176	11.183	1.733
8	0.144	1.932	10.255	7.891
9	2.41	1.412	12.057	7.392
10	5.042	0.797	12.674	20.755
11	2.989	2.809	11.516	26.216
12	14.5	4.903	0.988	2.294
13	11.149	1.207	5.192	0.53
14	3.736	5.841	9.211	25.563
15	0.818	8.494	12.022	20.307
16	0.636	14.538	15.122	7.086
Mean	4.37	4.554	9.304	9.606
Standard deviation	4.085	4.843	5.447	8.801

For all the objects, the position errors along the z-axis of the end-effector frame have not been reported since they are negligible due to the object geometry. For the same reason, the orientation errors around the z-axis of the end-effector frame can be negligible for the cast iron crankcase oil separator cover and the air pipe.

In some tests, large errors have been experienced, mostly along the y-axis of the end-effector frame, but the object has been successfully grasped since the gripper is characterized by parallel fingers, and errors along the closing direction are more tolerated.

The system failures can be divided into two main categories:

1. Errors related to missing (see Figure 9a,e) or inaccurate (see Figure 9b,d,f) keypoint detection or prediction, and wrong depth estimation.
2. Pose estimation errors that can cause the slipping of the object.

Table 7. Mean errors for the plastic crankcase oil separator cover.

Successful Tests	$e^e_{p_x}$ [mm]	$e^e_{p_y}$ [mm]	$e^e_{\phi_x}$ [deg]	$e^e_{\phi_y}$ [deg]	$e^e_{\phi_z}$ [deg]
1	13.453	1.036	7.726	4.594	10.505
2	1.795	8.396	11.419	16.797	1.977
3	1.387	5.306	2.214	3.451	11.761
4	7.493	5.909	3.41	5.945	7.285
5	0.799	6.753	3.439	12.744	2.62
6	2.997	1.521	2.03	4.885	9.134
7	5.897	9.452	10.56	1.247	4.62
8	0.973	1.319	7.221	0.007	4.146
9	2.246	5.577	2.104	3.025	2.231
10	1.268	13.631	3.325	4.165	26.108
11	4.547	13.786	2.265	1.418	7.61
12	0.575	1.472	8.638	2.531	10.629
13	3.606	2.318	1.721	6.289	23.799
Mean	3.618	5.883	5.082	5.161	9.417
Standard deviation	3.487	4.281	3.381	4.527	7.367

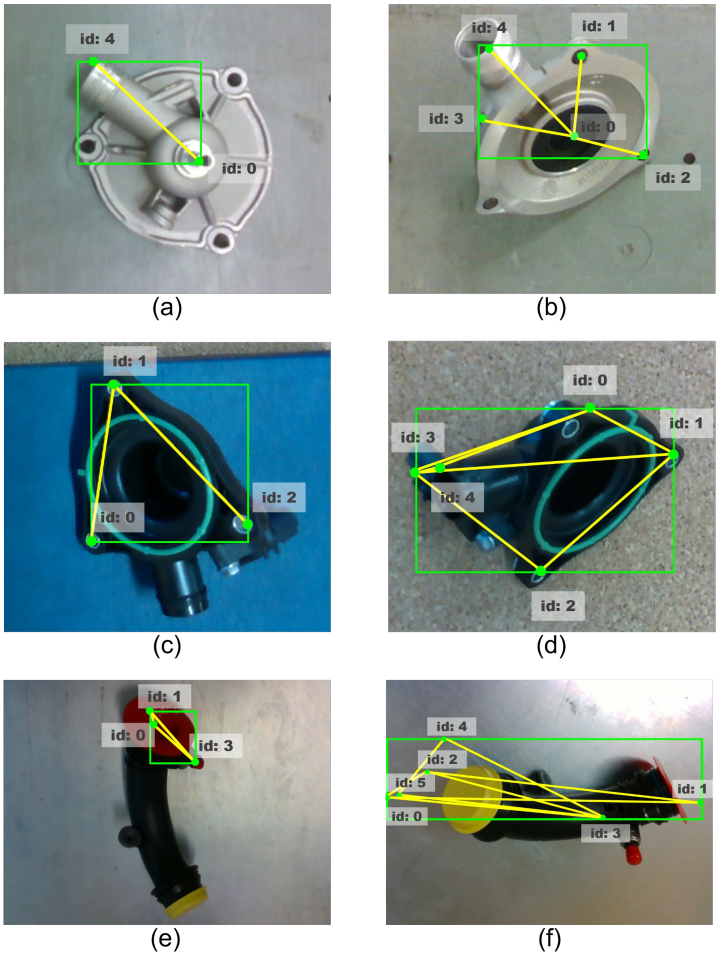

Figure 9. Examples of missing (**a**,**e**) and inaccurate (**b**,**d**,**f**) keypoint detection in TS. Example of missing keypoint detection that can lead to a successful object grasping (**c**).

In the case of a missing keypoint detection, e.g., due to the relative object-camera position, the failure can be managed by moving the camera's point of view and acquiring a new prediction (see Section 3.3). In the other cases, the grasping procedure is completed with a failure. Since the robot can detect the grasping failure, the whole process is repeated.

It is worth noticing that, according to the procedure outlined in Algorithm 2, the grasping of an object is feasible even with only three visible keypoints (see Figure 9c) correctly detected, provided that one of these is a grasping point situated in a location that can be grasped with the available end-effector.

5. Conclusions

In this work, a robust method for complex-geometry parts grasping in an industrial scenario has been proposed. In such an environment, grasping challenges are due to the presence of uncertainties in the position of the object to grasp and to the perception of noise due to its material. In particular, we focused on real-world automotive parts with complex geometries and reflective surfaces that provoke noise in the depth map. The proposed solution relies on a TS extraction network that can create a graph-based representation of the object in real time. A reasoning step is used to decide if the current view of the object is good enough for the actual grasping or if the manipulator needs to move to better grasp the object. Quantitative experiments have been conducted with a 7 DoF robot and three different complex-shaped automotive parts, demonstrating that the proposed approach is fast and robust. The high accuracy and real-time capability of this proposed approach render it a suitable solution for industrial applications where fast and accurate performance is required.

Due to the complexity of the considered objects, a complete quantitative performance comparison with other approaches present in the literature can hardly be carried out. However, the test dataset is publicly available to make possible future comparisons.

In future directions, we intend to study the integration of the depth data inside the TS extraction process to directly obtain the 3D positions of the keypoints. Moreover, the object detection phase could also be integrated into the TS extraction procedure.

Author Contributions: Conceptualization, A.P., M.S., D.D.B. and F.P.; Methodology, A.P., M.S., D.D.B. and F.P.; Software, A.P., M.S. and D.D.B.; Investigation, A.P. and M.S.; Validation, A.P. and M.S.; Formal analysis, A.P., M.S. and D.D.B.; Writing—original draft, A.P. and M.S.; Writing— review and editing, D.D.B. and F.P.; Supervision, D.D.B. and F.P.; Project administration, D.D.B. and F.P.; Funding acquisition, F.P. All the authors contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Italian Ministry of University and Research under the grant PRIN 2022 PNRR MELODY (Multi-robot collaborativE manipuLation suppOrting DisassemblY tasks) n. P2022XALNS.

Data Availability Statement: The source code of the TS detector approach is publicly available at https://github.com/apennisi/CoGP-TS. The ROS-based source code of our approach is publicly available at https://github.com/sileom/graspingWithSkeleton.git. Several videos of the experiments are available at https://tinyurl.com/bdhyf493. The test set images are available at https://tinyurl.com/bdxs8n7z. All the models used in the described strategy are publicly available and can be downloaded from https://tinyurl.com/3a4nnc88 (All links accessed on 23 July 2024).

Acknowledgments: The authors would like to thank Alessandro Lorenzo, Simona D'Amato, and Antonio Giardiello for their help with the image annotation process.

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

Abbreviations

The following abbreviations are used in this manuscript:

CNN	Convolutional Neural Network
DoF	Degree of Freedom
FCI	Franka Control Interface
FCN	Fully Convolutional Network
FN	False Negative

FP	False Positive
GG-CNN	Generative Grasping Convolutional Neural Network
MBConv	Mobile inverted Bottleneck Convolution
OKS	Object Keypoint Similarity
PAF	Part Affinity Fields
ROS	Robot Operating System
TN	True Negative
TP	True Positive
TS	Topological Skeleton

References

1. Sileo, M.; Bloisi, D.D.; Pierri, F. Real-time Object Detection and Grasping Using Background Subtraction in an Industrial Scenario. In Proceedings of the 2021 IEEE 6th International Forum on Research and Technology for Society and Industry (RTSI), Virtual, 6–9 September 2021; pp. 283–288.
2. Sileo, M.; Bloisi, D.D.; Pierri, F. Grasping of Solid Industrial Objects Using 3D Registration. *Machines* **2023**, *11*, 396. [CrossRef]
3. Costanzo, M.; De Maria, G.; Lettera, G.; Natale, C. Can robots refill a supermarket shelf?: Motion planning and grasp control. *IEEE Robot. Autom. Mag.* **2021**, *28*, 61–73. [CrossRef]
4. Asif, U.; Tang, J.; Harrer, S. GraspNet: An Efficient Convolutional Neural Network for Real-time Grasp Detection for Low-powered Devices. In Proceedings of the IJCAI, Stockholm, Sweden, 13–19 July 2018; pp. 4875–4882.
5. Zhang, H.; Tan, J.; Zhao, C.; Liang, Z.; Liu, L.; Zhong, H.; Fan, S. A fast detection and grasping method for mobile manipulator based on improved faster R-CNN. *Ind. Robot. Int. J. Robot. Res. Appl.* **2020**, *47*, 167-175. [CrossRef]
6. Mahler, J.; Liang, J.; Niyaz, S.; Laskey, M.; Doan, R.; Liu, X.; Ojea, J.A.; Goldberg, K. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. In Proceedings of the Robotics: Science and Systems (RSS), Cambridge, MA, USA, 12–16 July 2017.
7. Pinto, L.; Gupta, A. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In Proceedings of the 2016 IEEE international conference on robotics and automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 3406–3413.
8. Schmidt, P.; Vahrenkamp, N.; Wächter, M.; Asfour, T. Grasping of unknown objects using deep convolutional neural networks based on depth images. In Proceedings of the 2018 IEEE international conference on robotics and automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 6831–6838.
9. Morrison, D.; Corke, P.; Leitner, J. Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach. In Proceedings of the Robotics: Science and Systems (RSS), Pittsburgh, PA, USA, 26–30 June 2018.
10. Morrison, D.; Corke, P.; Leitner, J. Learning robust, real-time, reactive robotic grasping. *Int. J. Robot. Res.* **2020**, *39*, 027836491985906. [CrossRef]
11. Yu, F.; Koltun, V. Multi-scale context aggregation by dilated convolutions. *arXiv* **2015**, arXiv:1511.07122.
12. Dune, C.; Marchand, E.; Collowet, C.; Leroux, C. Active rough shape estimation of unknown objects. In Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and System, Nice, France, 22–26 September 2008; pp. 3622–3627.
13. Kraft, D.; Pugeault, N.; Başeski, E.; POPOVIĆ, M.; Kragić, D.; Kalkan, S.; Wörgötter, F.; Krüger, N. Birth of the object: Detection of objectness and extraction of object shape through object–action complexes. *Int. J. Humanoid Robot.* **2008**, *5*, 247–265. [CrossRef]
14. Detry, R.; Ek, C.H.; Madry, M.; Piater, J.; Kragic, D. Generalizing grasps across partly similar objects. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, St Paul, MI, USA, 14–18 May 2012; pp. 3791–3797.
15. Bloisi, D.D.; Pennisi, A.; Iocchi, L. Background modeling in the maritime domain. *Mach. Vis. Appl.* **2014**, *25*, 1257–1269. [CrossRef]
16. Marchand, É.; Spindler, F.; Chaumette, F. ViSP for visual servoing: A generic software platform with a wide class of robot control skills. *IEEE Robot. Autom. Mag.* **2005**, *12*, 40–52. [CrossRef]
17. Kannala, J.; Brandt, S.S. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Trans. Pattern Anal. Mach. Intell.* **2006**, *28*, 1335–1340. [CrossRef] [PubMed]
18. Tsai, R.Y.; Lenz, R.K. A new technique for fully autonomous and efficient 3 D robotics hand/eye calibration. *IEEE Trans. Robot. Autom.* **1989**, *5*, 345–358. [CrossRef]
19. Osokin, D. Real-time 2D Multi-Person Pose Estimation on CPU: Lightweight OpenPose. *arXiv* **2018**, arXiv:1811.12004.
20. Cao, Z.; Hidalgo, G.; Simon, T.; Wei, S.E.; Sheikh, Y. OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 172–186. [CrossRef] [PubMed]
21. Howard, A.; Sandler, M.; Chu, G.; Chen, L.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for MobileNetV3. *arXiv* **2019**, arXiv:1905.02244.
22. Siciliano, B.; Sciavicco, L.; Villani, L.; Oriolo, G. *Robotics—Modelling, Planning and Control*; Springer: London, UK, 2009.
23. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
24. Brooks, J. COCO Annotator. 2019. Available online: https://github.com/jsbroks/coco-annotator/ (accessed on 23 July 2024).

25. Ronchi, M.R.; Perona, P. Benchmarking and Error Diagnosis in Multi-Instance Pose Estimation. *arXiv* **2017**, arXiv:1707.05388.
26. Heartexlabs; Lin, T. LabelImg. 2015. Available online: https://github.com/heartexlabs/labelImg (accessed on 23 July 2024).