Received May 12, 2021, accepted May 28, 2021, date of publication June 4, 2021, date of current version June 14, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3086417

Memory Efficient Grasping Point Detection of Nontrivial Objects

PETR DOLEZEL[1], **(Member, IEEE), DOMINIK STURSA**[1], **(Member, IEEE), DUSAN KOPECKY**[1], **(Member, IEEE), AND JIRI JECHA**[2]
[1]Faculty of Electrical Engineering and Informatics, University of Pardubice, 53210 Pardubice, Czech Republic
[2]K2 Machine Ltd., 53002 Pardubice, Czech Republic

Corresponding author: Petr Dolezel (petr.dolezel@upce.cz)

This work was supported by the European Regional Development Fund (ERDF)/European Social Fund (ESF) "Cooperation in Applied Research between the University of Pardubice and companies, in the Field of Positioning, Detection and Simulation Technology for Transport Systems (PosiTrans)" under Grant CZ.02.1.01/0.0/0.0/17_049/0008394.

ABSTRACT Robotic manipulation with a nontrivial object providing various types of grasping points is of an industrial interest. Here, an efficient method of simultaneous detection of the grasping points is proposed. Specifically, two different 3 degree-of-freedom end effectors are considered for simultaneous grasping. The method utilizes an RGB data-driven perception system based on a specifically designed fully convolutional neural network called attention squeeze parallel U-Net (ASP U-Net). ASP U-Net detects grasping points based on a single RGB image. This image is transformed into a schematic grayscale frame, where the positions and poses of the grasping points are coded into gradient geometric shapes. In order to approve the ASP U-Net architecture, its performance was compared with nine competitive architectures using metrics based on generalized intersection over union and mean absolute error. The results indicate its outstanding accuracy and response time. ASP U-Net is also computationally efficient enough. With a more than acceptable memory size (77 MB), the architecture can be implemented using custom single-board computers. Here, its capabilities were tested and evaluated on the NVIDIA Jetson NANO platform.

INDEX TERMS Robotic grasping, grasping point detection, machine vision, deep learning, convolutional neural network.

I. INTRODUCTION

The number of operational industrial robots relative to the number of workers (i.e., robot density) is constantly growing. In 2020, the global robot density was 113 units per 10 000 employees. In Western Europe, it was even a greater number with 225 units per 10 000 employees [1]. Along with this growth, the requirements for quality, price and new features of industrial robots are also continuously increasing. The ability to grasp objects is one of the essential functionalities of modern industrial robots. Various approaches to robotic grasping have long been researched. According to Kumra and Kanan [2], the robotic grasping system is composed of the grasping point detection using a perception system, trajectory planning, and execution using a robotic arm. A demonstrative example of a robotic grasping system is depicted in Fig. 1. In this work, the first step of the robotic grasping system, i.e. grasping point detection, is targeted.

A perception system for the robotic grasping system is supposed to solve a visual recognition problem in order to detect graspable objects in a scene. The sensors applied to perceive the scene are typically 3-D vision sensors, RGB-D cameras or RGB cameras [3]. Consequently, the data for graspable object detection are in the form of point clouds, RGB-D maps or just classical RGB images. Nevertheless, the crucial part is to detect and locate possible grasping points in the scene from sensor information. In addition, the full information provided to the subsequent step (i.e. trajectory planning) must include the predicted position and pose of the robotic arm end effector.

Apparently, perception system designers deal with several important and interacting objectives. They need to select a suitable type of sensor, detection algorithm and supporting hardware. These choices depend on the scene arrangement (types, positions, and poses of objects, light conditions, dustiness, etc.), required minimal accuracy, required maximal response time of the system, and cost. Considering sensor types, 3-D vision sensors provide a very accurate 3-D map

The associate editor coordinating the review of this manuscript and approving it for publication was Yizhang Jiang.

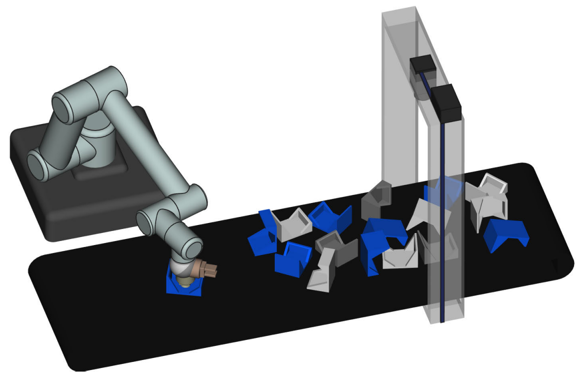


FIGURE 1. Example of a robotic stand with a perception system. A conveyor belt brings objects for manipulation, an industrial camera takes an RGB image of the scene, and the perception system determines grasping points. Lastly, the robotic arm, which is equipped with two kinds of end effectors, puts the objects into the desired positions.

of the scene, but they are expensive, and their frame-rate is rather low. RGB cameras are affordable, provide high resolution of the scene with high frame-rate, but only in 2D perspective. Hence, RGB cameras are especially applied for manipulation with objects scattered in a single-layer manner. Contrary to RGB cameras, RGB-D sensors provide a depth map of the scene. However, the resolution of the depth map is often too low to be used for accurate grasping point detection. Bearing in mind detection algorithms, a big difference is, whether the system deals with known or unknown objects, whether the objects are primitive (cylinders, cubes) or complex, whether the objects are scattered separately or in dense clutters, whether overlapping positions occur in the scene, etc. In addition, the supporting hardware needs to be determined in order to fulfill the computational complexity and required response time of the detection algorithm. While some methods can be deployed on simple and affordable single-board computers, others require full-weight industrial computers with high computational capabilities.

In this work, a novel grasping point detection algorithm usable in the perception system is proposed. The intention of the proposed method is a fast detection and localization of all feasible grasping points in the scene, which are suitable for defined end effectors of the robotic arm, namely, parallel grippers, and vacuum cups. The proposed method especially aims at nontrivial objects with several possible grasping points. These objects can be arranged in dense overlapping clutters. Since RGB data are considered as the input source of information, the method deals with arrangements, where objects are scattered in a single-layer manner. Furthermore, due to the RGB data, 3 degree-of-freedom end effectors (i.e. end effectors with free position and fixed pose) are used. From the practical point of view, 3 degree-of-freedom end effectors are used by a large family of selective compliance assembly robotic arms (SCARA). Although with limited number of degrees of freedom, SCARA robots are generally faster than comparable Cartesian robot systems and are therefore applied to many time critical applications. Naturally, they are offered by all main producers of industrial robots [4]–[6].

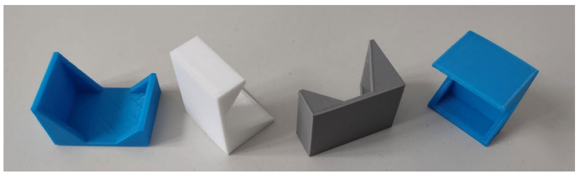


FIGURE 2. The object for manipulation (color variations). Each object provides various grasping points for manipulation, depending on its position and pose.

The key part of the proposed method is a specifically designed fully convolutional neural network called attention squeeze parallel U-Net (ASP U-Net). ASP U-Net transforms the scene with the objects into a schematic grayscale image, where the grasping points are highlighted with the gradient geometric shapes. In addition, the full information about the grasping point, necessary for the defined end effector, is explicitly coded in those gradient shapes. ASP U-Net can be deployed on selected single-board computers (especially targeted on NVIDIA Jetson NANO) with an acceptable response time, which increases the probability of application in industry.

As an example of the capabilities of the proposed method, the perception system for grasping point detection of specifically prepared nontrivial objects was implemented in this work. These objects provide various edges suitable for a parallel gripper and various planes for a vacuum cup - see Fig. 2. Random clutters of these objects can produce an infinite number of scene arrangements, where the feasibility of the grasping points is affected by mutual disposition of the objects. The authors believe that this example sufficiently demonstrates the capabilities of the proposed method.

The key contributions of this article are as follows:

- An efficient method of grasping point detection for manipulation using a robotic arm is proposed.
- The method is generally applicable for simultaneous grasping point detection, and aims at various 3 degree-of-freedom end effectors, i.e. end effectors with free position and fixed pose (grippers, vacuum cups, screw drivers, etc.)
- An enhanced approach to transformation of the RGB image of the scene into a schematic grayscale image is introduced. The positions and poses of the grasping points are coded into gradient geometric shapes.
- The transformation is based on the specifically designed fully convolutional neural network ASP U-Net. The network is computationally efficient enough to be applied using custom single-board computers.

II. RELATED WORK

Over the past decades, various approaches to the robotic grasping have been introduced. Categorization of these approaches can be performed according to many different criteria. Principally, the robotic grasping method can be developed using an analytic or data-driven procedure. Analytic approaches examine the geometric shape of the target object to recognize possible grasp poses. Data-driven (empirical) methods, on the other hand, utilize machine learning, i.e. they extract the information from sample data in order to detect the grasp pose without having to be analytically programmed [7]. The following paragraphs are focused on a brief categorization of data-driven methods, since they have, in most cases, already been shown to outperform the analytic approaches to the grasping point detection in terms of computational complexity and accuracy [8], [9]. For analytic approaches, the surveys of various methods can be found in [10], [11].

The data-driven approaches can be divided into model-based and model-free methods [12]; or according to manipulated object type (rigid, deformable, articulated, etc.) [13]. Significant criterion is whether the approach is capable of handling unknown objects, or operates only with known objects [14]. Furthermore, the approaches can be designed for different kinds of sensors (RGB, RGD-D, ToF sensors, laser scanners, or sensor fusion) [15], [16], and for various types of end effectors (parallel gripper, three-finger gripper, vacuum cup, or even custom-made gripper) [17]. Moreover, some approaches deal only with grasping of a single separated object, while others are able to handle grasping in a dense clutter [18]. A comprehensive review of various categorizations of data-driven approaches can be found in the recent work of Kleeberger *et al.* [9].

Considering the industrial environment with changing conditions, highly variable surroundings and dynamical production plans, very flexible but robust methods to robotic grasping are needed. Specifically, data-driven model-free approaches capable of handling multiple objects are required due to industrial demand. Most recent academic contributions to this area use deep convolutional networks [19] and even some industrial robotics producers have begun to implement deep convolutional networks-based methods in their commercial applications [20].

Deep convolutional networks can be used for a grasping point detection in two alternative ways. They can either be used to provide an estimation of the object position and pose, and subsequent evaluation of grasping points, or they can provide the estimation of possible grasping point position in a single step. Apparently, the first alternative provides more data about the scene and enables utilization of this information in various applications, such as quality control. However, the second alternative is much more computationally efficient and therefore faster.

A. DEEP CONVOLUTIONAL NETWORKS FOR OBJECT POSITION AND POSE ESTIMATION

In recent years, convolutional network-based approaches have been a leading methodology for the object position and pose estimation. The most apparent input for position and pose estimation is a 3D map or a depth map of the scene, delivered by a point cloud and an RGB-D snapshot, respectively. Considering these inputs, a broad set of approaches has been introduced. For example, the OP-Net was presented in [21]. The OP-net is a fully convolutional neural network trained on synthetic data. Sock *et al.* [22] proposed the architecture capable of jointly learning multiple subtasks (2D detection, depth and 3D pose estimation), as well as joint registration of multiple objects. Another fully convolutional neural network was the PPR-Net introduced by Dong *et al.* [23]. This network estimated a position and pose for each point in a point cloud of the object instance to which it belongs. Then, the density-based clustering procedure was applied, and the resulting position and pose was obtained by averaging the predicted information for each cluster.

Since sensors providing a 3D map of the scene are still expensive, several approaches aim to offer the object position and pose estimation using a single or a sequence of RGB images. As an example, Kehl *et al.* [24] presented a detector for 3D instance detection and full pose estimation based on SSD architecture [25]. The Deep-6DPose [26] implemented another well-known architecture called R-CNN [27]. Another approach was introduced in [28]. Authors here decoupled the position and pose estimation based on a prior segmentation step, and they considered both RGB and RGB-D input data.

B. DEEP CONVOLUTIONAL NETWORKS FOR DIRECT GRASPING POINT ESTIMATION

Direct grasping point estimation is expected to be generally more time efficient than the object position and pose estimation with a consecutive grasping point identification in an estimated model of the object. In [29], one of the first convolutional neural network-based applications for a direct grasping point estimation was introduced. The authors focused on the detection of grasping points suitable for parallel grippers, and they also provided a comprehensive dataset of grasping points with various suitability. Moreover, in [30], the same dataset was used for real-time application. Similarly, authors in [31] also dealt with a grasping point detection for a parallel gripper, and they proposed a fully convolutional neural network, which was based on ResNet architecture [32]. In all the mentioned sources, the grasping point detector worked in a sliding window way. Hence, the computational complexity was strongly affected by the number of windows. Another mutual attribute was that the grasping point was always represented by a rectangle, which indicated the point, pose, and opening of the parallel gripper. The mentioned methods were able to correctly predict the grasping points of scattered objects and displayed acceptable generalization capabilities. These methods, however, often failed to work with cluttered objects [19].

As an alternative to representation of the grasping point as a rectangle, different approaches provided pixel-wise performed transformations of the scene. These transformations should represent an affordance of the robotic arm to grasp the object in a particular point. One of the first works [33] let the robot iteratively pick and drop objects to get a continuous density function. More recent contributions often implement convolutional neural networks. In [34], the authors provided a method which directly labels the areas affordable to grip. These areas then needed to be transformed into specific information for a gripper. A multipart solution for the grasping point detection was introduced in [35]. Here, the authors considered a multifunctional gripper (vacuum cup and parallel gripper in various poses) and they predicted the grasping score for each type of gripping. Another pixel-wise affordance-based method, which provided not only grasping points, but general surfaces for robotic manipulation, was introduced in [36]. Lastly, authors in [37] proposed an affordance interpreter network to predict a pixelwise affordance map. Contrary to the above-mentioned methods, this map provided several regions with various data about grasping (stable horizontal grasp points, negative grasp locations, background, etc.). However, they did not consider different end effectors.

C. DEEP CONVOLUTIONAL NETWORKS FOR SEMANTIC SEGMENTATION

Semantic segmentation is an indispensable step in many aforementioned approaches. The objective is always to transform the input image into a representation that is easier to analyse. According to [38], deep learning-based approaches to image semantic segmentation can be divided into more than 10 groups. In this contribution, selected example architectures, which have already proven to be efficient with some specific tasks, are listed.

One of the first proposed deep learning-based works for image semantic segmentation was the Fully Convolutional Network (FCN) [39]. The authors modified existing architectures by replacing fully connected layers with convolutional layers. As the result, their model provided a spatial segmentation map instead of classification scores.

Nowadays, most of the convolutional neural network architectures for semantic segmentation are based on convolutional encoder-decoder procedure. The SegNet architecture [40] consists of an encoder network, which is topologically identical to the 13 convolutional layers in the VGG16 network [41], and a corresponding decoder network followed by a pixel-wise classification layer. This approach was similarly used with ResNet [32], MobileNet [42] or DenseNet [43] as a backbone. As a part of encoder-decoder family of approaches, U-shape architectures attract special attention. They refine their structure into the U-shape network structure using deconvolutional or up-sampling layers. The U-Net [44] introduced the skip connection network structure as well, and the BiSeNet [45] brought two parallel paths in its architecture. A spatial path was expected to preserve the spatial information and generate high-resolution features, while a context path was employed to obtain a sufficient receptive field. Apart from that, attention mechanisms were successfully applied to semantic segmentation problems, where same objects occurred in different scales, with different contrast or in different context. As an example, authors in [46] proposed an attention gate to an existing U-Net architecture and proved a considerable rise of the prediction performance contrary to the original U-Net. Lastly, considering edge computing, some authors tried to reduce the number of parameters of the existing architectures while keeping the performance. Beheshti *et al.* proposed the Squeeze U-Net [47], which reduced the number of parameters from 30 million (original U-Net) to 2.59 million with the same accuracy using a CamVid road scenes dataset [48].

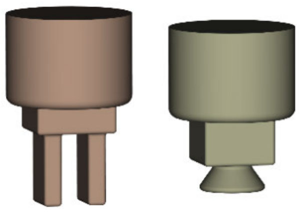


FIGURE 3. Considered end effectors. The parallel gripper (left) requires the information about the position of the grasping point and the suitable angle for grasping. The vacuum cup (right) requires the information about the position of the grasping point only.

III. METHOD STATEMENT

A. PROBLEM FORMULATION

Robotic manipulation with a nontrivial object providing various types of grasping points is a complex task. A typical industrial assignment consists of multiple, randomly placed, oriented and cluttered objects on a conveyor belt (see Fig. 1). The shapes of the objects are known in advance, but the positions and poses are not. The perception system of an employed robotic stand must be fast and accurate, enabling a quick decision about the position of the grasping point and an automatic replacement of the appropriate end effector at the same time.

The herein proposed solution consists of an RGB data-driven perception system which detects grasping points based on a single RGB image of the scene and provides all the necessary information for the 3 degree-of-freedom end effector.

In order to keep the study concise, the following experiments were limited to two types of end effectors; the parallel gripper and the vacuum cup (see Fig. 3). Moreover, the method targets on objects scattered in a single-layer manner. The undeniable advantage of this arrangement is the unnecessity of depth information, since all grasping points occur in a similar vertical distance from the sensor. Hence, RGB data can be considered as sufficient input source of information. The nontrivial object for manipulation was defined as shown in Fig. 2. The object provides two kinds of grasping points – edge and plane – for manipulation using either a parallel gripper or a vacuum cup, depending on the object position and pose.

B. OBJECT POSES

The RGB perception system is intended for the grasping point detection of the objects scattered randomly on a flat surface. The above-mentioned conveyor belt transports the objects to the robotic arm for further manipulation. Clearly, various arrangements of the objects in the scene can occur, and each individual arrangement presumably provides a specific set

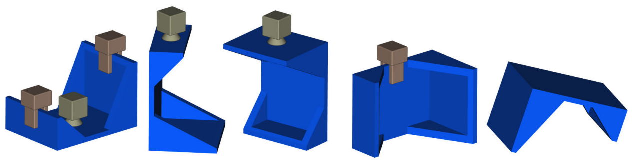


FIGURE 4. Grasping points of the object at various poses. All possible grasping points are highlighted with a parallel gripper symbol and with a vacuum cup symbol. Note that the object provides two opposite rectangularly shaped edges and another four triangularly shaped edges. However, the triangularly shaped parts of the objects are not considered to provide a grasping point robust enough to be applied neither with the parallel gripper (left image) nor with the vacuum cup (second image from the right). These parts do not meet the minimal dimension requirements. Hence, these points are not marked as possible grasping points. Moreover, note that the pose of the object situated in the fifth image does not provide any grasping point at all.

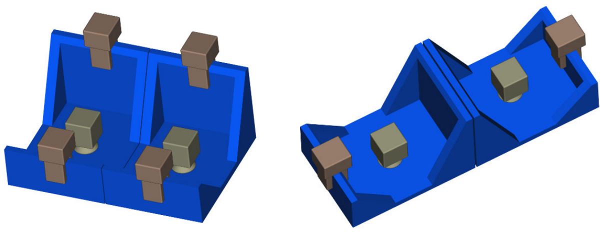


FIGURE 5. Grasping points of two adjacent objects. Note that in the second case, the adjacent edges do not allow use of the parallel gripper to grasp the objects.

of possible grasping points. For this study, considering end effectors in Fig. 3, the following requirements are stated for the grasping points to be feasible. Specifically, for the vacuum cup (planes), the free circle with diameter equal to or greater than 12 mm needs to be available. For the parallel gripper, the edge with length equal to or greater than 12 mm is necessary to be accessible. Furthermore, a free rectangular place with dimensions equal to or greater than 10 mm from each side of the edge is required for the parallel gripper to be able to grab the object.

Discussing a single object in a scene, five different poses and up to two grasping points for the parallel gripper, and zero or one grasping point for the vacuum cup can be obtained. All possible poses are figured in Fig. 4.

A more complex arrangement occurs with two or more objects in the scene. In this case, available grasping points are affected not only by the pose of the current object, but also by a position and pose of the subordinate object. Two possible arrangements of two closely adjacent objects are situated in Fig. 5. While the contact using lateral edges does not affect the grasping points, the front contact removes the expected grasping point.

Other arrangements are caused by irregular contacts of the objects including mutual overlaps. In some cases, the grasping point space is reduced, while in extreme cases the grasping points are completely removed. Selected examples of object arrangements are shown in Fig. 6.

As a conclusion of the arrangements stated above, it is expected that the proposed RGB perception system will

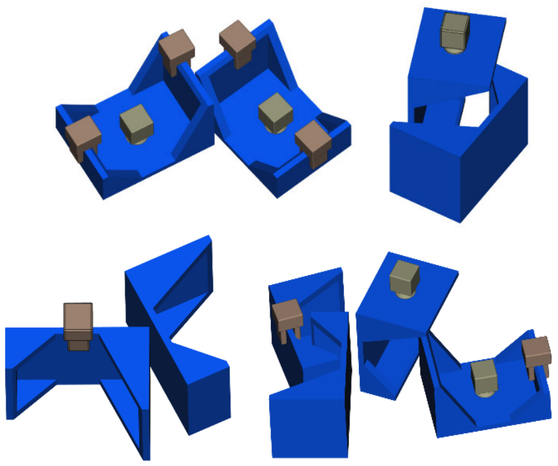


FIGURE 6. Grasping points of randomly situated objects with irregular contacts or mutual overlaps (selected examples). All feasible grasping points are highlighted. Note that some grasping points theoretically meet the minimal dimension requirements, but they are not feasible for a 3 degree-of-freedom end effector since they are covered by another object (top-right and bottom-right example). However, these grasping points may become feasible after removal of the covering object.

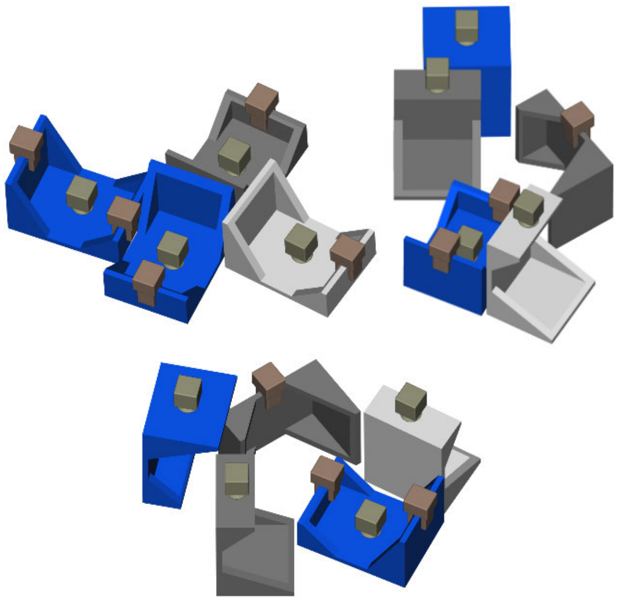


FIGURE 7. Grasping points of randomly situated objects with irregular contacts or mutual overlaps (full scene examples). All feasible grasping points are highlighted.

successfully detect combinations of grasping points in scenes as shown in Fig. 7.

C. FULLY CONVOLUTIONAL NEURAL NETWORK FOR GRASPING POINT DETECTION

Current state-of-the-art image processing systems are based on deep convolutional networks (ConvNets) [49]. Recently introduced ConvNets allow the design of a full image processing system which ensure image pre-processing, feature extraction, region-of-interest localization and classification. The most popular topologies are based on YOLO architecture [50] and SSDNets [25]. However, a different branch of ConvNets has also been applied to the image processing – especially for the semantic image segmentation. Briefly, the image segmentation is a task, where specific regions in an image are labelled according to defined characteristics. Specifically, each pixel of the image is labelled with a corresponding class. The objective is to simplify or change the image into a representation that is more meaningful and easier to analyse. Fully convolutional neural networks, such as U-Net [44], FCN [39], or SegNet [40] provide their capability to be especially successful in this area.

1) GRASPING POINT REPRESENTATION

The task is to detect and locate two types of the grasping points in a scene (i.e. edge or plane) in order to recognize specific grasping regions of an image which are applicable for the parallel gripper or for the vacuum cup. One option is to transform this task into a semantic image segmentation-like problem. Specifically, each pixel in an RGB image representing the scene would be labelled with a float number in the range $< 0; 1 >$, where 1 means the optimal and 0 means the most unsuitable grasping point position. Since two types of grasping points are considered, two different labels for each pixel of the input image are required – the first one for the parallel gripper and the second one for the vacuum cup. In addition, the desired spatial orientation of the parallel gripper should be included in the labels of surrounding pixels. Therefore, the label of each pixel was determined considering the parallel gripper as a function of the distance of the pixel from two end points of the abscissa. Both, the grasping point and the angle of the parallel gripper are defined by this approach. Specifically, for pixel x

$$\text{label}_x = \left(\frac{d_{12}}{d_{1x} + d_{2x}}\right)^a \frac{1}{1 + b{d_{0x}}^c}, \tag{1}$$

where d_{12} is the length of the abscissa, i.e. the distance between points, which define the grasping point; d_{1x} is the distance between the current pixel and the first end point of the abscissa; d_{2x} is the distance between the current pixel and the second end point of the abscissa; d_{0x} is the distance between the current pixel and the middle point of the abscissa. Parameters a, b, c affect the size of the shape and need to be set according to the features of the scene.

Using the aforementioned transformation, the spatial representation of the grasping point for a parallel gripper is obtained, where the optimal position is labelled with 1 and the label value decreases with the distance from it. However, the steepest descent goes perpendicularly from the abscissa. On the other hand, the descent of the label value is gradual in parallel to the abscissa. This situation is highlighted in Fig. 8.

Apparently, the optimal grasping point is defined by a label equal to 1, while the required spatial orientation (pose) of a parallel gripper is defined by the direction of the most gradual descent of the label values. This grasping point representation directly provides all necessary information for a robotic arm to manipulate the object.

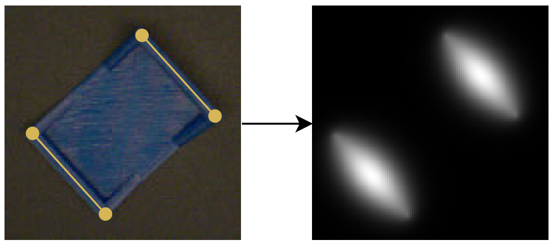


FIGURE 8. Grasping point representation for a parallel gripper. The grasping point is represented by a yellow abscissa in the original RGB image (left); the end points are highlighted by circles. Labels of each pixel calculated according to eq. (1) are represented by intensities (right); the higher the intensity, the closer the label is to 1. The black color is equal to 0. Considering the presented representation, the optimal position of the grasping point is defined by the center of the gradient shape (i.e. maximal intensity), and the required angle (pose) of the parallel gripper is defined by the most gradual descent of the intensity.

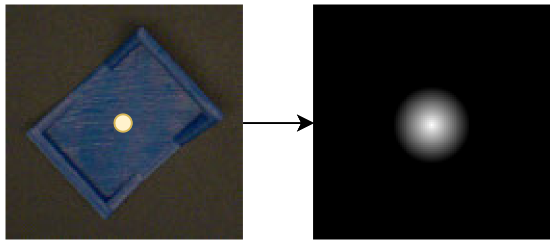


FIGURE 9. Grasping point representation for a vacuum cup. The grasping point is represented by a yellow circle in the original RGB image (left). Labels of each pixel calculated according to the eq. (2) are represented by intensities (right); the higher is the intensity, the closer is the label to 1, black color is equal to 0. Considering the presented representation, the optimal position of the grasping point is defined by the center of the gradient circle (i.e. maximal intensity).

Considering a vacuum cup, the situation is simpler, since the information about the spatial orientation is not required. Therefore, the label of each pixel is determined as a function of the distance of the pixel from the optimal position of the grasping point. Specifically, for pixel x

$$\begin{aligned} \text{label}_x &= \frac{R - d_{0x}}{R} \quad \text{for} \quad d_{0x} < R, \\ \text{label}_x &= 0 \;\; \text{for} \quad d_{0x} \geq R, \end{aligned} \tag{2}$$

where d_{0x} is the distance between the current pixel and the optimal position of a grasping point, and R is the parameter representing the radius of the shape. The situation is shown in Fig. 9.

Thus, the presented neural network is intended to transform the original RGB image of the scene into two schematic grayscale images, where the grasping points are highlighted as gradient shapes, which can efficiently provide all the necessary information for a robotic arm. The outputs of the network are schematically depicted in Fig. 10.

2) ASP U-Net ARCHITECTURE

The presented ASP U-Net neural network is based on a ConvNet topology capable of performing a transformation as seen in Fig. 10. In addition, since the presented perception system should be implemented in real-time industrial applications using single-board computer architectures, neural

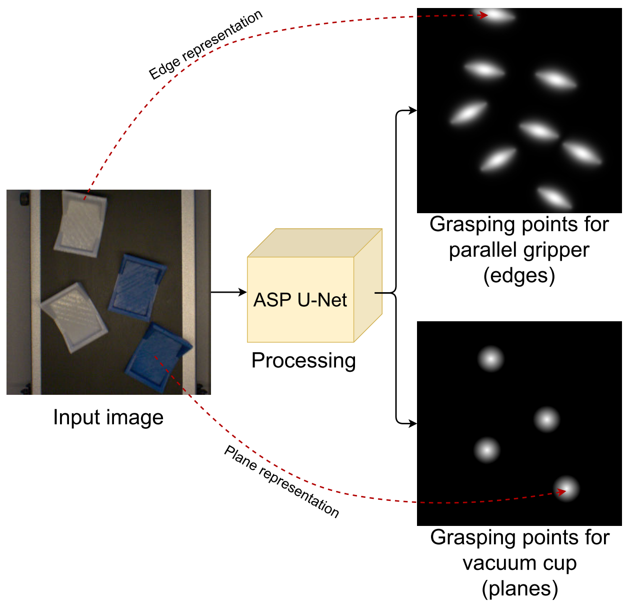

Edge representation
Input image
ASP U-Net
Processing
Grasping points for parallel gripper (edges)
Plane representation
Grasping points for vacuum cup (planes)

FIGURE 10. ASP U-Net is intended to transform the original RGB image into a pair of schematic images, where the positions of the grasping points are highlighted as gradient shapes. Namely, the first output from ASP U-Net should represent the positions and poses of the grasping points suitable for a parallel gripper (i.e., edges), and the second output should represent the positions of the grasping points suitable for a vacuum cup (i.e., planes).

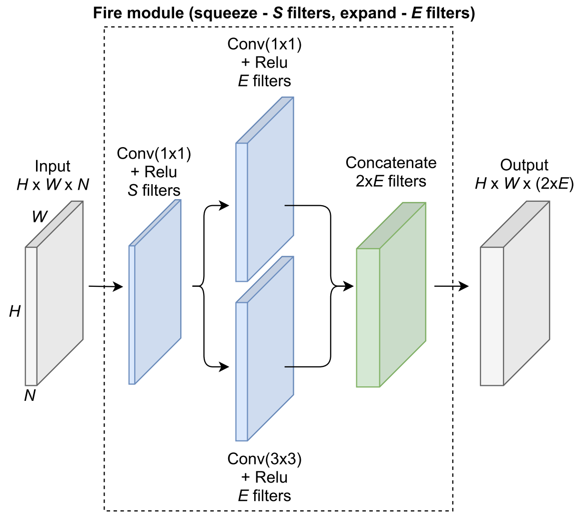


FIGURE 11. Fire module. Input tensor dimensions are labelled with W, H, N. S and E are the numbers of filters in the modules.

network size and resulting demands on computing power are considered, too.

To accomplish this goal, U-Net was used as an initial architecture [44]. U-Net is a U-shaped fully convolutional neural network proposed initially for biological and medical image segmentation tasks. However, it has since been adapted to many other applications. It follows a typical encoder-decoder architecture with a bottleneck. Nevertheless, it also includes direct signals from the encoder part to the decoder part which allow the network to propagate context information to higher resolution layers. U-Net has proven itself to be

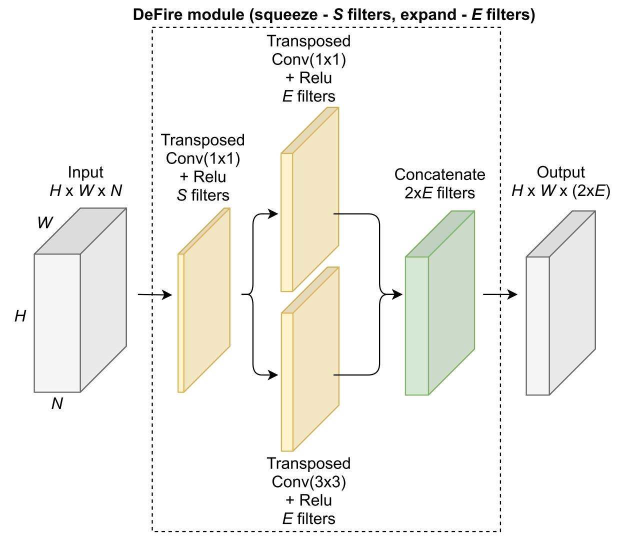


FIGURE 12. DeFire module. Input tensor dimensions are labelled with W, H, N. S and E are the numbers of filters in the modules.

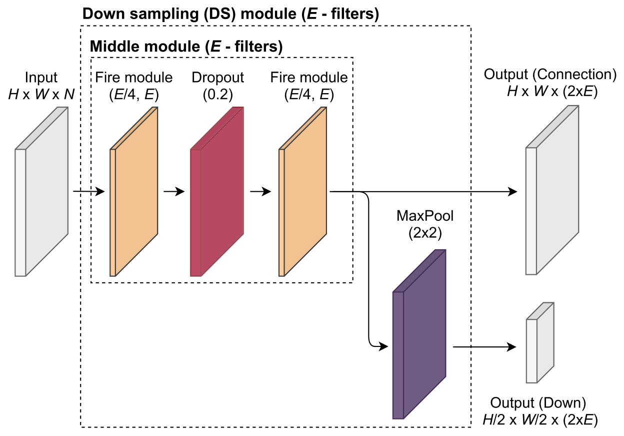


FIGURE 13. Down sampling module with middle module. Input tensor dimensions are labelled with W, H, N. The middle module processes input by Fire modules connected through a dropout layer. The down sampling module provides two outputs. The connection output is taken directly from the middle module output, and its width W and height H dimensions are the same as the input. The down output is obtained by processing the middle module output through a max pooling layer, which halves the input width and height. For both output layers, the filter count is defined by doubling the number of parameterized expand filters E.

very successful in semantic image segmentation, especially in cases where only small training datasets are available [51]. This feature is very beneficial for embedded applications. However, U-Net is defined by more than 30 million parameters with a size of 364 MB in memory.

Therefore, inspired by SqueezeNet [52] and SqueezeSegNet [53], classical convolutional and transposed convolutional layers were replaced with layers similar to Fire module and DeFire module, respectively. See Fig. 11 and Fig. 12 for a detailed explanation. Fire and DeFire modules were implemented into Down sampling module (Fig. 13) and Up sampling module (Fig. 14) in the encoder and decoder parts of U-Net. According to [47], these replacements can provide more than 10× parameter reduction while keeping the accuracy.

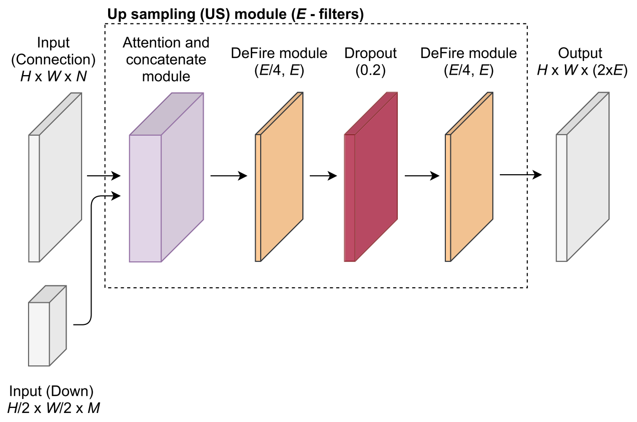

Up sampling (US) module (*E* - filters)
Input (Connection) *H* x *W* x *N*
Attention and concatenate module
DeFire module (*E*/4, *E*)
Dropout (0.2)
DeFire module (*E*/4, *E*)
Output *H* x *W* x (2x*E*)
Input (Down) *H*/2 x *W*/2 x *M*

FIGURE 14. Up sampling module. Two input tensors, where width *W* and height *H* of the down input must be half of the connection input width *W* and height *H*, are concatenated by the attention and concatenate module. Then, the signal is processed by two DeFire modules connected through a dropout layer. Resulting output *W* and *H* dimensions are the same as for the connection input, and the filter size is double that of the parameterized expand filters *E*.

Another improvement is based on the fact, that the aim of the network is to label grasping points located on the objects of several colors under varying light conditions. Hence, the attention mechanism based on an attention gate, as introduced in [46], is implemented. Attention gates are situated in the decoder part of the architecture (see Fig.15) and they filter the features propagated through the skip connections.

Lastly, the network should generate two images, each of them providing the information about one type of the grasping point. Due to this fact, it would be beneficial to parallelize the propagation of data through the network at some point, in order to provide separate parts of the architecture for edge processing and for plane processing, respectively. Therefore, the U-part of the original U-Net architecture was replicated and mirrored. Both parallel parts were concatenated at the concluding section of the network. With this enhancement, the network is expected to adapt its parameters to process the edge-relevant features in one parallel branch, and the plane-relevant features in the other branch.

The resulting ASP U-Net (abbrev. of attention squeeze parallel U-Net) refers to the attention mechanism, parameter reduction, parallel detection of different types of grasping point, and U-Net origin. The architecture is depicted in Fig. 16 and the TensorFlow-Keras implementation of the architecture is publicly available at [54].

3) COMPETITIVE ARCHITECTURES

In order to approve the ASP U-Net architecture, its performance was compared with some of the state-of-the-art architectures. Specifically, SegNet [40], BiSeNet [45], U-Net [44], and FCN-VGG16 [39] were implemented according to their original configurations. Additionally, Squeeze U-Net [47] and Attention U-Net [46] were included for comparison, in order to follow the recent trends of edge computing and attention mechanisms. Lastly, some semantic segmentation neural networks based on classical convolutional neural networks as backbones were added into consideration. Namely, ResNet 101 [32], DenseNet 121 [55], and MobileNet [42] in combination with FCN architecture were implemented. These architectures were referred to as FCN-ResNet101, FCN-DenseNet121 and FCN-MobileNet, respectively.

All the architectures were adapted to work with the same data, i.e. the input layer and the output layer for each architecture were replaced with same layers as in the ASP U-Net architecture.

D. TRAINING DATASET ACQUISITION

In order to acquire training and testing data, a demonstrative robotic stand was prepared – see Fig. 17.

A Basler acA2500-14uc industrial RGB camera [56], as the RGB sensor, was implemented. This sensor is able to provide up to 14 5-MPx RGB frames per second. The camera was equipped with a Computar M3514-MP lens [57] in order to monitor the scan area of 300 $\times$ 420 mm from a distance of 500 mm.

1) DATA COLLECTION

Initially, in total 716 images were captured for a training set. The resulting collection of images included from 0 to 9 objects of three colors (blue, white, grey) with various positions and poses. Many images contained only parts of the objects. Resolution of the images was 288 $\times$ 288 RGB pixels with 8-bit depth.

2) DATA LABELLING

Each image of the training set has to be labelled in order to be usable for neural network training. Specifically, the positions of the grasping points have to be determined and consequently, the grayscale schematic images according to Fig. 8 and Fig. 9 have to be prepared for each image. Hence, a custom labelling application (GraspLabeller), capable of manual preparation of the required data, was programmed. GraspLabeller allows the marking of grasping points using a computer mouse and it provides the resulting pair of grayscale images, as seen in Fig. 18. The labelling procedure needs to be performed in accordance with the minimal dimension requirements for a parallel gripper and for a vacuum cup, as defined in the paragraph III-B.

Considering the image resolution and the overall scene arrangement, the parameters in eqs. (1) and (2) were set as follows: $a = 10$, $b = 0.002$, $c = 2$, $R = 20$.

3) DATA AUGMENTATION

Various data augmentation techniques are often used in computer vision to enhance the datasets. In order to use only the label-preserving augmentation methods, the geometric transformations of the original RGB images and the target pairs of grayscale images were used. Specifically, the original dataset was enhanced in the following way.

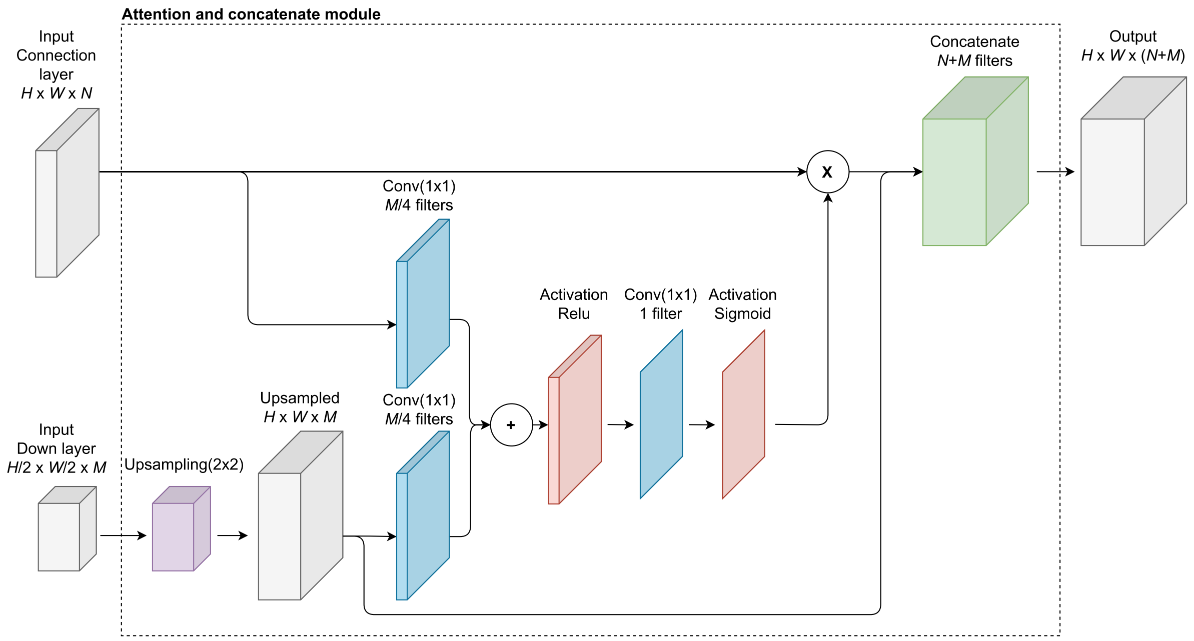


FIGURE 15. **Attention and concatenate module. This module is composed from two parts. The attention part, where features from the upsampled down layer input and connection layer input are passed through the convolutional layer with filter size one quarter of down layer input filter size *M* and summed for further processing. After summation, the relu activation function is used as an input to the convolutional layer with 1 filter, which is then processed with a sigmoidal activation function and multiplied with the original input connection layer. The concatenate part joins the connection layer after the attention process with an upsampled down layer, and produces output with *H*, *W* size the same as in the connection layer and the sum of both input layer filters.**

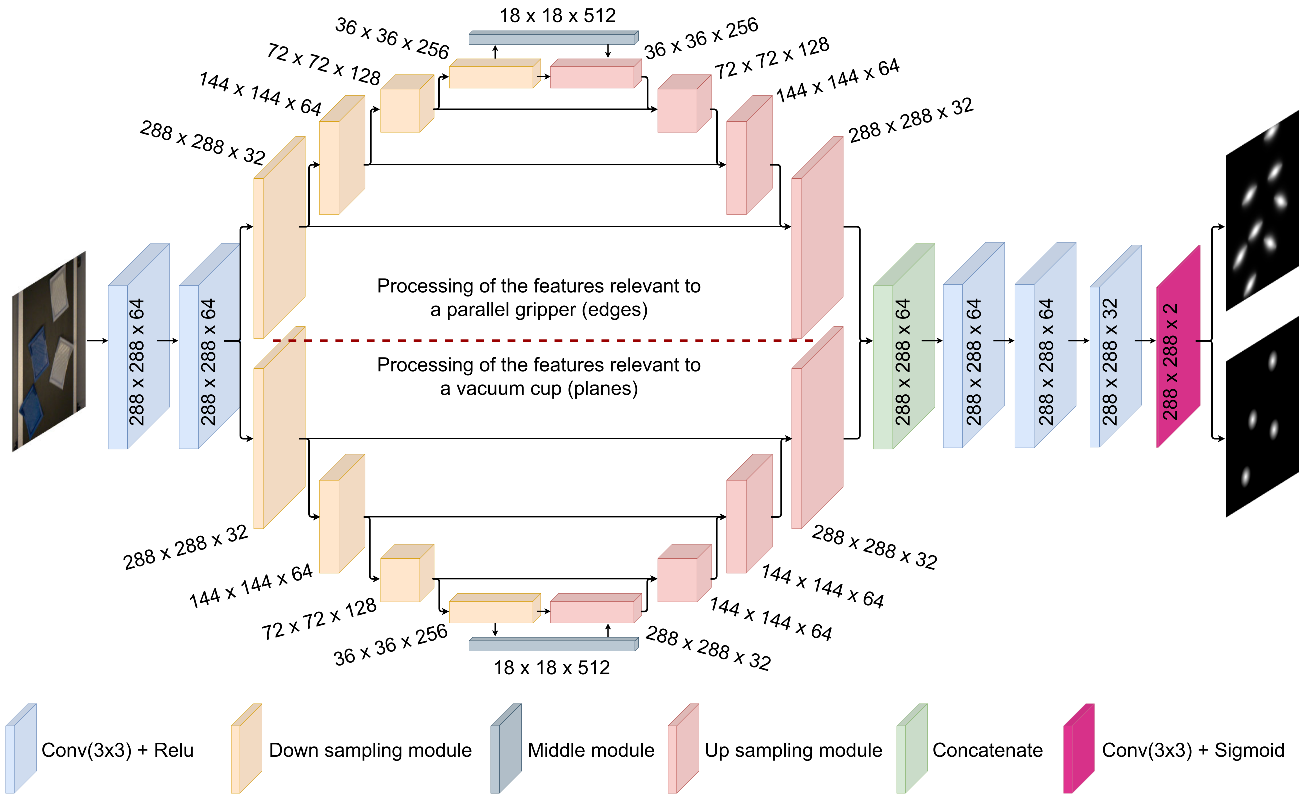


FIGURE 16. **ASP U-Net. An input image is processed by two convolutional layers with 64 filters and relu activation function. Then, the signal is split into two parallel branches, each for processing different features – edges and planes. Each branch is composed of four gradually interconnected down sampling modules, which are also connected to four up sampling modules in the same depth. A bottleneck is represented by the middle module, which connects the deepest down and up sampling modules. Both branches are then concatenated and processed with two convolutional layers with 64 filters, and one convolutional layer with 32 filters, all using the relu activation function. The final convolutional layer, with the sigmoidal activation function and 2 filters, provides output schematic images.**

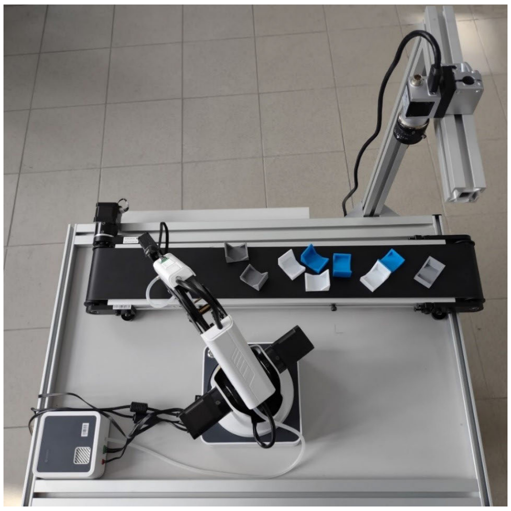


FIGURE 17. A demonstrative robotic stand for data acquisition. Objects of interest are scattered on a conveyor belt. An industrial camera situated above the scene is used for RGB image acquisition. A line LED light is situated in the aluminium profile near the camera.

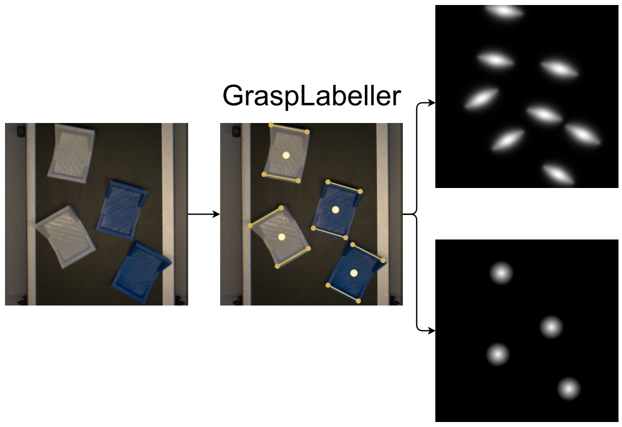


FIGURE 18. Dataset labelling using GraspLabeller. Each grasping point in every image is manually labelled using a custom application and transformed into a pair of grayscale schematic images.

- Each sample was randomly rotated by an angle between $\langle -10^\circ, 10^\circ\rangle$ with an even distribution of probability.
- Each sample was shifted between $\langle -10\text{ px}, 10\text{ px}\rangle$ with an even distribution of probability in both horizontal and vertical direction.

Therefore, the original dataset of 716 images was enhanced to 2148 images.

4) DATASET FOR TESTING

Another dataset is required to evaluate the perception system. Generally, some fraction of the training dataset is put aside and used for this purpose. However, a manual acquisition of the dataset was performed in this case. Hence, an additional sequence of object arrangements was created and acquired. All possible setups were considered for inclusion into the testing set. Furthermore, the difficult and troublesome distributions of the objects (i.e. various contact positions and onerous attainability of the grasping points) were specifically selected to be involved. In total, 54 unique image samples were acquired for the testing set. These images contain 148 objects with 236 available grasping points, where 143 of them are suitable for parallel gripper and the rest of them for vacuum cup.

The whole dataset is publicly available at [54].

TABLE 1. Parameters of the training.

Input shape	288 x 288 x 3
Training algorithm	Adam optimizer
Number of experiments	5
Number of samples	2148
Validation split	0.3
Initialization	Normal distribution (mean = 0, std = 0.05)
Number of epochs	300
Criterion for resultant model	Loss function value over validation set
Learning rate α	0.001
Exponential decay rate 1 β_1	0.9
Exponential decay rate 2 β_2	0.999

E. NEURAL NETWORK TRAINING

ASP U-Net, as well as the competitive state-of-the-art architectures, were trained using an Adam optimizer, since it is generally considered to provide satisfactory performance in most cases [58], [59]. Initial weights were set randomly with Gaussian distribution. Binary cross entropy was used as the loss function. A measure of 30 % of the dataset was put aside as the validation set. The training experiments were performed five times for each architecture in order to reduce the stochastic character of the training, i.e. to prevent the loss function being stuck in a local minimum. The best instances, considering the loss function over the validation set, were then evaluated. All the parameters of the training are summarized in Table 1.

F. EVALUATION METRIC

After training of all the considered architectures, each network was evaluated. From the accuracy point of view, a well-known IoU metric (intersection over union) was adapted. However, this metric evaluates a detector based on ground-truth bounding boxes and predicted bounding boxes. The proposed perception system did not provide a sharp bounding box, but a grayscale image, where the predicted position depended on the pixel intensity. Hence, the generalized IoU (gIoU) metric was defined as follows.

$$\begin{array}{c}\text{Area}\\\text{of}\\\text{Overlap}\end{array} = \begin{bmatrix} o_{11} & o_{12} & \dots & o_{1N} \\ o_{21} & o_{22} & \dots & o_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ o_{M1} & o_{M2} & \dots & o_{MN} \end{bmatrix}, \tag{3}$$

$$\begin{array}{c}\text{Area}\\\text{of}\\\text{Union}\end{array} = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1N} \\ u_{21} & u_{22} & \dots & u_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ u_{M1} & u_{M2} & \dots & u_{MN} \end{bmatrix}, \tag{4}$$

$$o_{ij} = \min(t_{ij}, y_{ij}) \text{ for } i = 1, \dots, M, \; j = 1, \dots, N, \tag{5}$$

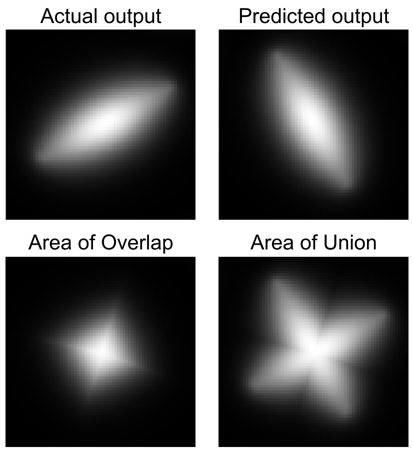


FIGURE 19. **Demonstration of the generalized IoU metric. In this case,** gIoU = **0.4604.**

$$u_{ij} = \max(t_{ij}, y_{ij}) \text{ for } i = 1, \dots, M, \; j = 1, \dots, N, \tag{6}$$

$$\text{gIoU} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} o_{ij}}{\sum_{i=1}^{M}\sum_{j=1}^{N} u_{ij}}. \tag{7}$$

In equations above, t_{ij} is the value of the i^{th} row and j^{th} column of the target output, y_{ij} is the value of the i^{th} row and j^{th} column of the actual output, M is the number of rows and N is the number of columns in the output. See Fig. 19 for a graphical representation of the metric.

Since the proposed perception system provides two grayscale images as an output, the metric is evaluated separately for each output, referred as gIoU_e for edges and gIoU_p for planes.

A classical mean absolute error (MAE) metric is also implemented to interpret the results. For this case, the MAE is defined as follows.

$$\text{MAE} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} \left|t_{ij} - y_{ij}\right|}{M\,N}. \tag{8}$$

Again, the metric is evaluated separately for each output, referred as MAE_e for edges and MAE_p for planes.

Evaluation of the memory size and response time of the perception system is also important, because it is intended to be used in real-time industrial applications using single-board computer architectures. From this point of view, both, the size of the neural network and its response time were evaluated.

G. TECHNICAL IMPLEMENTATION

Experiments were carried out using the following hardware specification: Processor Intel Core i5-8600K (3.6 GHz), internal memory 16 GB DDR4 (2666 MHz), video card NVIDIA PNY Quadro P5000 16 GB GDDR5 PCIe 3.0 (2560 CUDA cores), SSD SATA M.2 512 GB. The experiments are performed using Python 3.6 and TensorFlow 2.0. In addition, the resulting perception system was deployed to an NVIDIA Jetson NANO single-board computer in order to test the response time, since it is a generally accepted benchmark system for embedded artificial intelligence-based applications [60]. This developer kit offers the Quad-core ARM A57 1.43 GHz CPU together with 4 GB RAM and it provides wide communication possibilities (USB 2.0, 3.0, SATA, WiFi).

Response times are referred to as T_{Work} for the desktop computer described above and as T_{NANO} for the NVIDIA Jetson NANO.

TABLE 2. **Final values of loss function over the validation set and time per epoch during training. Note that the competitive architectures were trained using the same procedure as ASP U-Net with the same dataset - see Section III-E.**

Architecture	Architecture adapted from	Best value	Worst value	Time per epoch, s
ASP U-Net	Proposed	0.03873	0.03932	145
Attention U-Net	[46]	0.03887	0.03915	105
BiSeNet	[45]	0.03784	0.03816	690
FCN-DenseNet121	[39], [55]	0.03769	0.03787	244
FCN-VGG16	[39]	0.03898	0.03928	119
FCN-MobileNet	[39], [42]	0.09677	0.30601	44
FCN-ResNet101	[39], [32]	0.03864	0.03889	81
SegNet	[40]	0.03842	0.03863	263
Squeeze U-Net	[47]	0.03852	0.03886	16
U-Net	[44]	0.03827	0.03852	93

IV. EXPERIMENTAL RESULTS

The proposed ASP U-Net, as well as the competitive architectures, were trained and validated five times according to the procedure addressed in Section III-E, with the dataset described in Section III-D. To show the training results, the best and the worst training session, according to the binary cross entropy loss function evaluated over the validation data at the end of the training session, were pointed out in Table 2. Furthermore, the average training times per epoch were also provided for each architecture.

In order to evaluate the effectiveness of ASP U-Net contrary to competitive architectures, the testing set described in Section III-D4 was used. Specifically, the metrics gIoU_e, gIoU_p, MAE_e and MAE_p, as described in Section III-F, were evaluated using the best-performing training session for each architecture. In addition, the memory size and the response time, using the workstation described in Section III-G and Jetson NANO were also considered for each selected architecture. The resulting values were summarized to Table 3.

The process of evaluation is depicted in Fig. 20. Firstly, each original input image from the testing set was propagated through the neural network. Then, the actual outputs were compared to their target representatives and the gIoU and MAE metrics were determined. After all samples from the testing sets were processed, the overall metrics were evaluated as the mean of the running values.

Apart from the overall metric values, some selected responses of ASP U-Net were expressed in Figs. 21 to 24. In all of the figures demonstrating quantitative results, the sequence of the original input image and the combination of the input image with both predicted outputs

TABLE 3. Metrics over testing set.

Architecture	Architecture adapted from	$\mathrm{gIoU_e}$	$\mathrm{gIoU_p}$	$\mathrm{MAE_e}$	$\mathrm{MAE_p}$	Size, MB	T_{Work}, s	T_{NANO}, s
ASP U-Net	Proposed	0.8675	0.9016	0.003192	0.001103	77	0.058	0.81
Attention U-Net	[46]	0.8425	0.8730	0.003806	0.001449	374	0.048	1.18
BiSeNet	[45]	0.8204	0.8539	0.003407	0.001455	463	0.186	7.39
FCN-DenseNet121	[39], [55]	0.8285	0.8685	0.003147	0.001451	110	0.078	2.27
FCN-VGG16	[39]	0.7885	0.8031	0.003894	0.002102	933	0.064	1.83
FCN-MobileNet	[39], [42]	0.1806	0.4938	0.019114	0.006177	93	0.031	0.33
FCN-ResNet101	[39], [32]	0.8340	0.8573	0.003605	0.001586	513	0.050	1.11
SegNet	[40]	0.8598	0.8897	0.003416	0.001243	303	0.086	3.15
Squeeze U-Net	[47]	0.8124	0.7751	0.004277	0.002323	30	0.028	0.19
U-Net	[44]	0.8494	0.8582	0.003243	0.001535	364	0.048	1.02

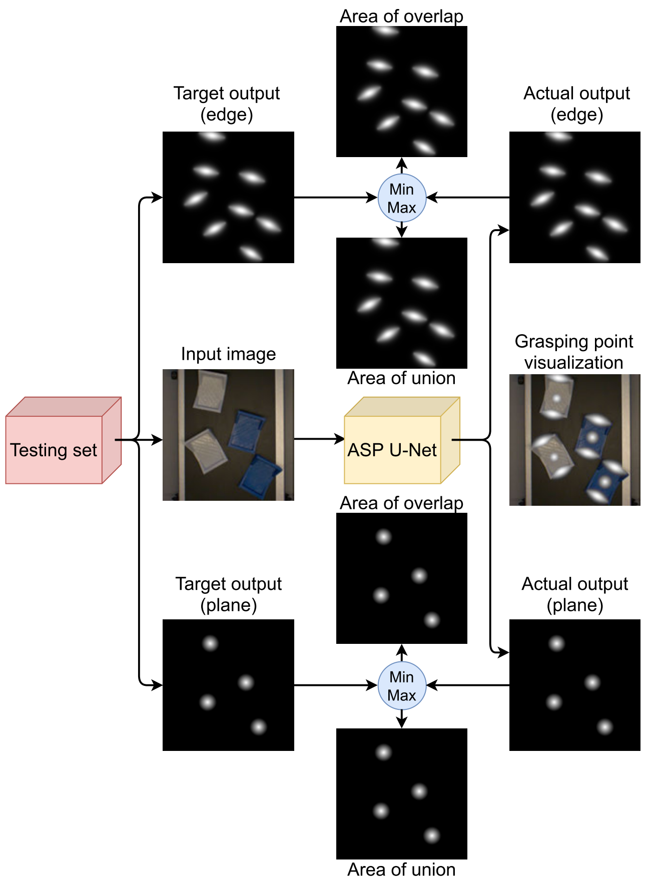


FIGURE 20. Response of ASP U-Net to a scene corresponding to Fig. 10 and Fig. 18. Actual outputs of ASP U-Net are compared to their expected variants and gIoU and MAE is computed. The outputs projected to the input image are shown as the grasping point visualization. In this case, $\mathrm{gIoU_e} = 0.9725$, $\mathrm{gIoU_p} = 0.9587$, $\mathrm{MAE_e} = 2.068 \times 10^{-3}$, $\mathrm{MAE_p} = 8.515 \times 10^{-4}$.

(edges and planes) was figured. In addition, the metric values were listed below the images.

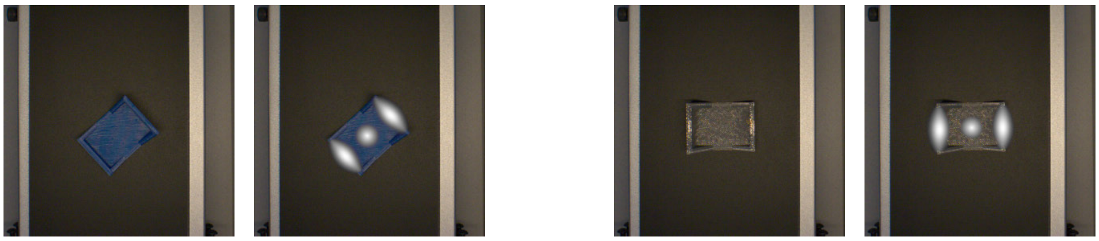
A. DISCUSSION

The proposed perception system, based on ASP U-Net architecture and a novel approach to pixel-wise transformation of the scene, dealt with the problem of the grasping point detection for the parallel gripper and the vacuum cup. All possible object positions and mutual interactions were considered, but the object cluttering was simplified to a single layer only. The proposed system did not directly address the problem of 3D orientation of the end effector. Hence, the system is suitable for the end effectors with a free position and fixed pose applied e.g. in the SCARA types of industrial robots.

According to the final values of the loss function during training, almost all considered architectures provided very similar results. Apart from FCN-MobileNet, where the training process failed to provide considerable results, the differences between best and worst results were insignificant. BiSeNet and FCN-DenseNet121 can be deduced as the most successful architectures, while the proposed ASP U-Net achieved average results. Training times differed significantly. While light-weight architectures (Squeeze U-Net, FCN-MobileNet) finished each epoch in tens of seconds, some other architectures approached five minutes per epoch, and BiSeNet even exceeded ten minutes. However, the training time is less important than the response time, since the training is in this case the once-and-for-all process.

However, the evaluated metrics over the testing set indicated a more significant variance of results. Considering $\mathrm{gIoU_e}$, $\mathrm{gIoU_p}$, $\mathrm{MAE_e}$ and $\mathrm{MAE_p}$, ASP U-Net provided the best results in three of them. BiSeNet and FCN-DenseNet121, which provided the best training results, were in many cases outperformed not only by ASP U-Net, but also by SegNet. A general trend could be observed, that the metrics defined for a parallel gripper grasping point detection provided worse values than the metrics defined for a vacuum cup. The partial exceptions are U-Net and Squeeze U-Net with similar results of $\mathrm{gIoU_e}$ and $\mathrm{gIoU_p}$.

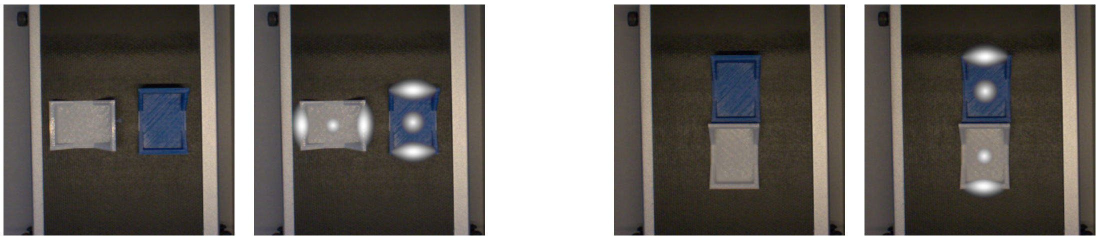
Considering the memory size and the response time, big differences between various architectures can be noticed. Memory efficient architectures, such as FCN-MobileNet or Squeeze U-Net, apparently need much less memory than full-weight architectures. Since the proposed ASP U-Net occupies less than 80 MB, it can be included in the group of memory efficient architectures. Response times using the workstation and Jetson NANO provided a similar trend. Although the response of ASP U-Net is slower than Squeeze U-Net, Attention U-Net and U-Net, using a desktop computer, it is still above average. ASP U-Net response time using Jetson NANO is even more favourable. It provides the best



$\mathrm{gIoU_e} = 0.8990$, $\mathrm{gIoU_p} = 0.9070$
$\mathrm{MAE_e} = 1.959 \times 10^{-3}$, $\mathrm{MAE_p} = 4.895 \times 10^{-4}$

$\mathrm{gIoU_e} = 0.9254$, $\mathrm{gIoU_p} = 0.8355$
$\mathrm{MAE_e} = 1.546 \times 10^{-3}$, $\mathrm{MAE_p} = 9.062 \times 10^{-4}$

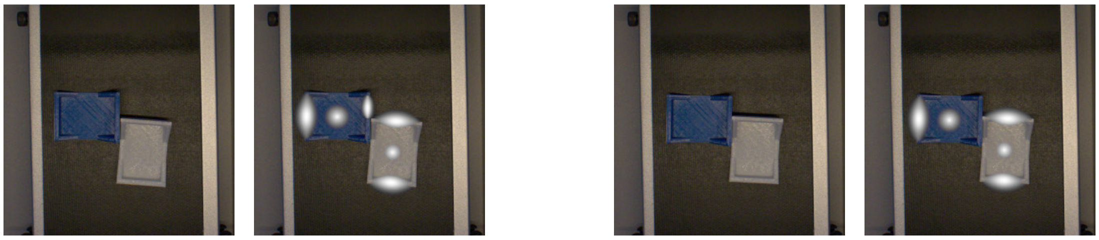
FIGURE 21. Response of ASP U-Net to a scene with one object. Two pairs of the input image and the concatenation of the original image with neural network output.



$\mathrm{gIoU_e} = 0.9289$, $\mathrm{gIoU_p} = 0.8619$
$\mathrm{MAE_e} = 2.899 \times 10^{-3}$, $\mathrm{MAE_p} = 1.494 \times 10^{-3}$

$\mathrm{gIoU_e} = 0.8984$, $\mathrm{gIoU_p} = 0.8115$
$\mathrm{MAE_e} = 2.015 \times 10^{-3}$, $\mathrm{MAE_p} = 2.118 \times 10^{-3}$

FIGURE 22. Response of ASP U-Net to a scene with two objects. The pair on the left represents the situation, where the feasibility of grasping points is not affected. In the image on the right, the adjacent edges do not allow use of the parallel gripper. Therefore, those grasping points are not highlighted.



$\mathrm{gIoU_e} = 0.8817$, $\mathrm{gIoU_p} = 0.8908$
$\mathrm{MAE_e} = 2.970 \times 10^{-3}$, $\mathrm{MAE_p} = 1.167 \times 10^{-3}$

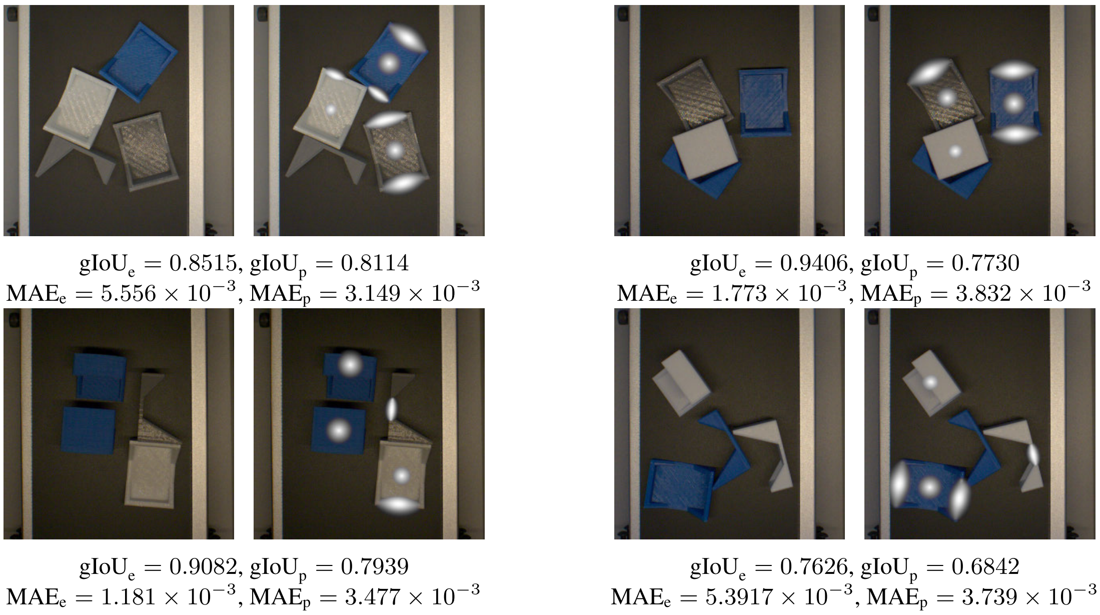
$\mathrm{gIoU_e} = 0.9206$, $\mathrm{gIoU_p} = 0.8582$
$\mathrm{MAE_e} = 2.383 \times 10^{-3}$, $\mathrm{MAE_p} = 1.536 \times 10^{-3}$

FIGURE 23. Response of ASP U-Net to a scene with two irregularly situated objects. Note that in the left photo, the grasping point space of the affected grasping point (blue object, right edge) is reduced from its original size because of the close position of the white object. However, the considered parallel gripper would still fit in the reduced place. In the right photo, where the white object is situated further upwards, the grasping point is completely rejected, because in this case, the parallel gripper would not fit in that place at all (i.e. the grasping point does not meet the minimal dimension requirements).

response time, except for the very light-weight architectures Squeeze U-Net and FCN-MobileNet.

The quantitative results presented in Table 3 and the example Figs. 20 - 24 show that ASP U-Net performs well with all recognized spatial orientations of the objects. This architecture can make an exact difference between feasible and unfeasible grasping points according to the spatial orientation of the objects in the scene. This phenomenon is considered to be one of the most significant features of the presented approach. Moreover, time response of the ASP U-Net is acceptable to be implemented in real-time applications. In other words, the proposed ASP U-Net provides competitive results to various full-weight state-of-the-art architectures in terms of accuracy, and it is still applicable for edge computing applications due to its low memory consumption and an exceptional response time.

The output of ASP U-Net provides information about the position of all feasible grasping points in the scene, and about the required pose of the end effector in the case of the parallel gripper. However, many objects may have some areas which are strongly preferable to grasp over others. For example, a knife could be grasped by its blade, but it would be better to grasp it by its handle. This situation can be dealt with either at the data labelling level (undesirable regions are not labelled



$\mathrm{gIoU_e} = 0.8515$, $\mathrm{gIoU_p} = 0.8114$
$\mathrm{MAE_e} = 5.556 \times 10^{-3}$, $\mathrm{MAE_p} = 3.149 \times 10^{-3}$

$\mathrm{gIoU_e} = 0.9406$, $\mathrm{gIoU_p} = 0.7730$
$\mathrm{MAE_e} = 1.773 \times 10^{-3}$, $\mathrm{MAE_p} = 3.832 \times 10^{-3}$

$\mathrm{gIoU_e} = 0.9082$, $\mathrm{gIoU_p} = 0.7939$
$\mathrm{MAE_e} = 1.181 \times 10^{-3}$, $\mathrm{MAE_p} = 3.477 \times 10^{-3}$

$\mathrm{gIoU_e} = 0.7626$, $\mathrm{gIoU_p} = 0.6842$
$\mathrm{MAE_e} = 5.3917 \times 10^{-3}$, $\mathrm{MAE_p} = 3.739 \times 10^{-3}$

FIGURE 24. Response of ASP U-Net to a scene with a group of randomly situated objects.

as the grasping points in the dataset), or by the implementation of some parent decision algorithm, which is intended to select the most suitable grasping point from the pool of possible ones. Moreover, the parent decision algorithm must be prepared for dealing with various extreme situations. Typically, no feasible grasping point can be detected in the current scene. Such a situation can be tackled by repeatedly evaluating the scene, or rearranging the objects in the scene by shaking or vibrating. Another situation is a presence of indistinctive shapes in the output image (e.g., with the peak intensity smaller than 0.50). In this case, a priority list of the detected grasping points needs to be set, and the robotic arm should select the most suitable grasping points in the first place. On all accounts, the parent decision system shall be prepared for the failure of the grasping procedure and it must be able to repeat it with actualized scene information.

The presented ASP U-Net architecture is expected to propagate the input signal into two parallel branches in order to perform edge processing and plane processing in a separate way. To demonstrate this phenomenon, the feature maps, gained from the last upsampling modules (see the upsampling modules 288 × 288 × 32 in Fig. 16) before the final concatenation, were visualized. A selected group of feature maps from the upper branch and from the lower branch is depicted in Fig. 25. The feature maps clearly indicate, that the features relevant to the parallel gripper (i.e., edges) are processed in the upper branch of the architecture, while the features relevant to the vacuum cup (planes) are processed in the lower branch of the architecture.

Considering the dataset created in this work, it is rather small in comparison to publicly available machine learning datasets. It is promising information, that proposed ASP U-Net shows good generalization capabilities. Specifically,

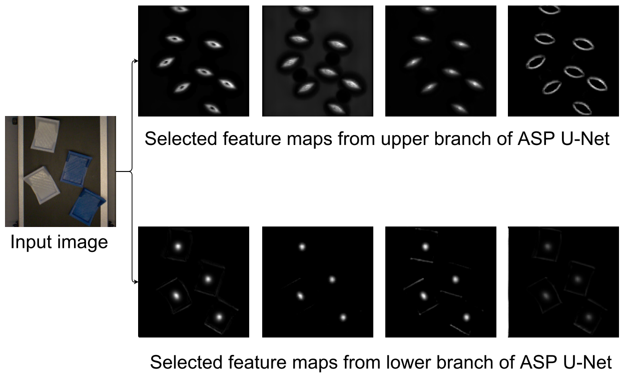

Selected feature maps from upper branch of ASP U-Net
Input image
Selected feature maps from lower branch of ASP U-Net

FIGURE 25. Selected feature maps provided by the last upsampling modules before the final concatenation. The input image corresponds to the images used in Figs. 10, 18 and 20. Note that the features relevant to the parallel gripper (i.e. edges) are processed in the upper branch of the architecture, while the features relevant to the vacuum cup (i.e. planes) are processed in the lower branch of the architecture.

while it provides average results during training, it exhibits the best results evaluated over the testing set, compared to all the selected competitive architectures. As a part of future work, the authors plan to verify the proposed ASP U-Net using the generally accepted publicly available grasping point datasets, such as the Extended Cornell Grasping Dataset [29]. It is also apparent, that the proposed perception system can be easily extended to work with RGB-D input data instead of RGB. With this extension, the feasibility of the grasping points could be decided not only from the spatial orientation of the objects, but also from the information about depth.

V. CONCLUSION

In this work, an innovative fully convolutional neural network architecture, called ASP U-Net, was proposed in order to be used in a robotic grasping system. ASP U-Net is based on the U-Net, but is improved by an attention mechanism based on an attention gate, and by Fire module, which reduces the number of parameters. In addition, the U-Net architecture was mirrored and concatenated with the signal from both parallel U-parts in order to extract different features parallelly from one input image.

ASP U-Net is generally applicable for simultaneous grasping point detection with various types of robotic arm end effectors. In this contribution, this architecture was comprehensively tested to detect and locate grasping points for a parallel gripper and a vacuum cup, using a novel pixel-wise performed transformation of the scene. Specifically, the positions and poses of the grasping points were coded into gradient geometric shapes, which efficiently described all the necessary information for a robotic arm to manipulate the objects. The performance of ASP U-Net was compared to nine competitive architectures and the results indicated an outstanding accuracy with more than acceptable memory size and response time.

REFERENCES

[1] International Federation of Robotics. (2021). *Robot Race: The World's Top 10 Automated Countries*. Accessed: Mar. 8, 2021. [Online]. Available: https://ifr.org/ifr-press-releases/news/robot-race-the-worlds-tops-10-automated-countries

[2] S. Kumra and C. Kanan, "Robotic grasp detection using deep convolutional neural networks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 769–776.

[3] G. Du, K. Wang, S. Lian, and K. Zhao, "Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: A review," *Artif. Intell. Rev.*, vol. 54, no. 3, pp. 1677–1734, Mar. 2021.

[4] Fanuc. (2021). *SCARA Robots for Increased Productivity*. Accessed: Apr. 20, 2021 [Online]. Available: https://www.fanuc.eu/uk/en/robots/robot-filter-page/scara-series

[5] Staubli. (2021). *Our SCARA Industrial Robots*. Accessed: Apr. 20, 2021. [Online]. Available: https://www.staubli.com/en/robotics/product-range/industrial-robots/4-axis-scara-robots/

[6] ABB. (2021). *IRB 910SC SCARA*. Accessed: Apr. 20, 2021 [Online]. Available: https://new.abb.com/products/robotics/industrial-robots/irb-910sc/

[7] A. Sahbani, S. El-Khoury, and P. Bidaud, "An overview of 3D object grasp synthesis algorithms," *Robot. Auton. Syst.*, vol. 60, no. 3, pp. 326–336, Mar. 2012.

[8] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis—A survey," *IEEE Trans. Robot.*, vol. 30, no. 2, pp. 289–309, Apr. 2014.

[9] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber, "A survey on learning-based robotic grasping," *Current Robot. Rep.*, vol. 1, no. 4, pp. 239–249, Dec. 2020.

[10] S. El-Khoury and A. Sahbani, "On computing robust n-finger force-closure grasps of 3D objects," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, Kobe, Japan, vols. 1–7, May 2009, p. 595.

[11] S. J. Dharbaneshwer, S. J. Subramanian, and K. Kohlhoff, "Robotic grasp analysis using deformable solid mechanics," *Meccanica*, vol. 54, nos. 11–12, pp. 1767–1784, Sep. 2019.

[12] C. Robinson, M. N. Saadatzi, and D. O. Popa, "Bin-picking using model-free visual heuristics and grasp-constrained imaging," in *Proc. IEEE 15th Int. Conf. Automat. Sci. Eng. (CASE)*, Aug. 2019, pp. 1618–1624.

[13] X. Wang, X. Jiang, J. Zhao, S. Wang, and Y.-H. Liu, "Grasping objects mixed with towels," *IEEE Access*, vol. 8, pp. 129338–129346, 2020.

[14] C. Gabellieri, F. Angelini, V. Arapi, A. Palleschi, M. G. Catalano, G. Grioli, L. Pallottino, A. Bicchi, M. Bianchi, and M. Garabini, "Grasp it like a pro: Grasp of unknown objects with robotic hands based on skilled human expertise," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2808–2815, Apr. 2020.

[15] R. Li and H. Qiao, "A survey of methods and strategies for high-precision robotic grasping and assembly tasks—Some new trends," *IEEE/ASME Trans. Mechatronics*, vol. 24, no. 6, pp. 2718–2732, Dec. 2019.

[16] C. Wang, X. Zhang, X. Zang, Y. Liu, G. Ding, W. Yin, and J. Zhao, "Feature sensing and robotic grasping of objects with uncertain information: A review," *Sensors*, vol. 20, no. 13, p. 3707, Jul. 2020.

[17] S. D'Avella, P. Tripicchio, and C. A. Avizzano, "A study on picking objects in cluttered environments: Exploiting depth features for a custom low-cost universal jamming gripper," *Robot. Comput.-Integr. Manuf.*, vol. 63, Jun. 2020, Art. no. 101888.

[18] C. Zhuang, Z. Wang, H. Zhao, and H. Ding, "Semantic part segmentation method based 3D object pose estimation with RGB-D images for bin-picking," *Robot. Comput.-Integr. Manuf.*, vol. 68, Apr. 2021, Art. no. 102086.

[19] S. Caldera, A. Rassau, and D. Chai, "Review of deep learning methods in robotic grasp detection," *Multimodal Technol. Interact.*, vol. 2, no. 3, p. 57, Sep. 2018.

[20] *IV2 Series Vision Sensor*, Keyence, Osaka, Japan, 2021. Accessed: Mar. 7, 2021.

[21] K. Kleeberger and M. F. Huber, "Single shot 6D object pose estimation," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2020, pp. 6239–6245.

[22] J. Sock, K. I. Kim, C. Sahin, and T.-K. Kim, "Multi-task deep networks for depth-based 6D object pose and joint registration in crowd scenarios," 2019, *arXiv:1806.03891*. [Online]. Available: https://arxiv.org/abs/1806.03891

[23] Z. Dong, S. Liu, T. Zhou, H. Cheng, L. Zeng, X. Yu, and H. Liu, "PPR-Net: Point-wise pose regression network for instance segmentation and 6D pose estimation in bin-picking scenarios," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 1773–1780.

[24] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1530–1538.

[25] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Proc. Eur. Conf. Comput. Vis.*, in Lecture Notes in Computer Science: Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, vol. 9905, 2016, pp. 21–37.

[26] T.-T. Do, M. Cai, T. Pham, and I. Reid, "Deep-6DPose: Recovering 6D object pose from a single RGB image," 2018, *arXiv:1802.10367*. [Online]. Available: https://arxiv.org/abs/1802.10367

[27] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[28] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," 2018, *arXiv:1711.00199*. [Online]. Available: https://arxiv.org/abs/1711.00199

[29] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *Int. J. Robot. Res.*, vol. 34, nos. 4–5, pp. 705–724, Apr. 2015.

[30] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2015, pp. 1316–1322.

[31] X. Zhou, X. Lan, H. Zhang, Z. Tian, Y. Zhang, and N. Zheng, "Fully convolutional grasp detection network with oriented anchor box," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 7223–7230.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[33] R. Detry, E. Baseski, M. Popović, Y. Touati, N. Krüger, O. Kroemer, J. Peters, and J. Piater, "Learning object-specific grasp affordance densities," in *Proc. IEEE 8th Int. Conf. Develop. Learn.*, Jun. 2009, pp. 1–7.

[34] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Detecting object affordances with convolutional neural networks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 2765–2770.

[35] A. Zeng *et al.*, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2018, pp. 3750–3757, doi: 10.1109/ICRA.2018.8461044.

[36] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Object-based affordances detection with convolutional neural networks and dense conditional random fields," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 5908–5915.

[37] J. Cai, H. Cheng, Z. Zhang, and J. Su, "MetaGrasp: Data efficient grasping by affordance interpreter network," in *Proc. Int. Conf. Robot. Automat. (ICRA)*, May 2019, pp. 4960–4966.

[38] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Feb. 17, 2021, doi: 10.1109/TPAMI.2021.3059968.

[39] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.

[40] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.

[41] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556

[42] G. A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: https://arxiv.org/abs/1704.04861

[43] S. Jegou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional DenseNets for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 1175–1183.

[44] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.*, in Lecture Notes in Computer Science: Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, vol. 9351, 2015, pp. 234–241.

[45] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "BiseNet: Bilateral segmentation network for real-time semantic segmentation," in *Proc. Eur. Conf. Comput. Vis.*, in Lecture Notes in Computer Science: Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, vol. 11217, 2018, pp. 334–349.

[46] O. Oktay, J. Schlemper, L. L. Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, S. McDonagh, N. Y. Hammerla, B. Kainz, B. Glocker, and D. Rueckert, "Attention U-Net: Learning where to look for the pancreas," 2018, *arXiv:1804.03999*. [Online]. Available: https://arxiv.org/abs/1804.03999

[47] N. Beheshti and L. Johnsson, "Squeeze U-Net: A memory and energy efficient image segmentation network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 1495–1504.

[48] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognit. Lett.*, vol. 30, no. 2, pp. 88–97, Jan. 2009.

[49] L. Jiao and J. Zhao, "A survey on the new generation of deep learning in image processing," *IEEE Access*, vol. 7, pp. 172231–172263, 2019.

[50] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[51] M. Bardis, R. Houshyar, C. Chantaduly, A. Ushinsky, J. Glavis-Bloom, M. Shaver, D. Chow, E. Uchio, and P. Chang, "Deep learning with limited data: Organ segmentation performance by U-Net," *Electronics*, vol. 9, no. 8, p. 1199, Jul. 2020.

[52] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 mb model size," 2016, *arXiv:1602.07360*. [Online]. Available: https://arxiv.org/abs/1602.07360

[53] G. Nanfack, A. Elhassouny, and R. O. H. Thami, "Squeeze-SegNet: A new fast deep convolutional neural network for semantic segmentation," *Proc. SPIE*, vol. 10696, Apr. 2018, Art. no. 106962O.

[54] P. Dolezel and D. Stursa. (2021). *Grasp Points Dataset for ASP U-Net Including the ASP U-Net Implementation*. Accessed: Mar. 26, 2021. [Online]. Available: https://www.researchgate.net/publication/350410726_Grasping_points_for_parallel_gripper_and_vacuum_cup_RGB_data

[55] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269.

[56] Basler. (2020). *Basler Ace*. Accessed: Jan. 8, 2021. [Online]. Available: https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca2500-14uc/

[57] Computar. (2020). *Computar Lenses*. Accessed: Jan. 8, 2021. [Online]. Available: https://computar.com/product/705/M3514-MP

[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, pp. 1–15, Dec. 2014.

[59] E. M. Dogo, O. J. Afolabi, N. I. Nwulu, B. Twala, and C. O. Aigbavboa, "A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks," in *Proc. Int. Conf. Comput. Techn., Electron. Mech. Syst. (CTEMS)*, Dec. 2018, pp. 92–99.

[60] A. A. Suzen, B. Duman, and B. Sen, "Benchmark analysis of jetson TX2, jetson nano and raspberry PI using deep-CNN," in *Proc. Int. Congr. Hum.-Comput. Interact., Optim. Robot. Appl. (HORA)*, Jun. 2020, pp. 1–5.

PETR DOLEZEL (Member, IEEE) received the Ph.D. degree from the University of Pardubice, Czech Republic, in 2009. Then, he defended his habilitation thesis at Tomas Bata University, in 2017, and he currently works as an Associate Professor and the Vice-Dean for research and development at the Faculty of Electrical Engineering and Informatics, University of Pardubice. His research interests include neural and evolutionary computation in process control, and signal and image processing. He is the author of more than 100 scientific contributions, including 20 journal articles and lectures at CORE ranked conferences. In addition, he has been a leader or member of research teams for a dozen research and development projects. As an academician, he led three Ph.D. students to a successful defence of their dissertations. He is a member of the technical program committee of several international conferences and is an active reviewer for numerous scientific journals. He intensively cooperates with research teams at the University of Burgos, Spain, and at the Slovak University of Technology, Slovakia.

DOMINIK STURSA (Member, IEEE) is currently pursuing the Ph.D. degree with the University of Pardubice, Czech Republic. His doctoral theses topic is an image processing applications with the deep neural networks. Since 2019, he has been an Employee and a Lecturer with the Department of Process Control, Faculty of Electrical Engineering and Informatics, University of Pardubice. He is a key member of a local research group led by P. Dolezel. He is an author of three journal articles and more than ten conference papers. His research is interested in fields of robotics, signal and image processing, and neural networks. He also has a membership with IEEE Robotics and Automation Society and IEEE Signal Processing Society. He cooperates with research team at the University of Burgos, Spain.

DUSAN KOPECKY (Member, IEEE) received the M.Sc. degree in measuring and control engineering in chemical and food industry and the Ph.D. degree in measuring technique from the University of Chemistry and Technology, Prague, Czech Republic, in 2006 and 2009, respectively. He is currently an Associate Professor of technical cybernetics with the University of Chemistry and Technology. He is also a Senior Researcher at the University of Pardubice, Czech Republic. He is an author of more than 40 articles in scientific journals and several books and chapters related to the field of measurement and control engineering. His research interest includes the development of physical sensors and new materials for shields against electromagnetic interferences.

JIRI JECHA received the M.S. degree in electrical engineering from the University of Pardubice, Pardubice, Czech Republic, in 2019.

He works as an Automation and Robotics Specialist at K2 Machine Ltd., Pardubice. His previous work experiences include PLC programming, Web development, and mechanical engineering. From 2017 to 2018, he completed an internship at the University of Southern Denmark, where he followed a programme focused on robotics, computer vision, artificial intelligence, and FPGA programming.