Robot arm grasping using learning-based template matching and self-rotation learning network

Minh-Tri Le1 · Jenn-Jier James Lien1

Received: 3 March 2022 / Accepted: 16 May 2022 / Published online: 6 June 2022
© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2022

Abstract
Applying deep neural network models to robot-arm grasping tasks requires the laborious and time-consuming annotation of a large number of representative examples in the training process. Accordingly, this work proposes a two-stage grasping model, in which the first stage employs learning-based template matching (LTM) algorithm for estimating the object position, and a self-rotation learning (SRL) network is then proposed to estimate the rotation angle of the grasping objects in the second stage. The LTM algorithm measures similarity between the feature maps of the search and template images which are extracted by a pre-trained model, while the SRL network performs the automatic rotation and labelling of the input data for training purposes. Therefore, the proposed model does not consume an expensive human-annotation process. The experimental results show that the proposed model obtains 92.6% when testing on 2400 pairs of the template and target images. Moreover, in performing practical grasping tasks on a NVidia Jetson TX2 developer kit, the proposed model achieves a higher accuracy (88.5%) than other grasping approaches on a split of Cornell-grasp dataset.

Keywords Self-supervised learning · Deep learning · Robot arm grasping · Template matching

1 Introduction
Deep learning is employed in an ever-increasing number of robotics applications nowadays [1–7]. Deep neural network (DNN) models offer significant advantages over traditional vision-based systems in the grasping and manipulation of objects affected by occlusion, illumination variations, reflection, and so on. In comparison with analytical methods based-robotic grasping applications using re-configurable grippers [8–10] in order to obtain flexible robot grasping, DNN models generally rely on affordance detection techniques [11, 12] or point cloud [13, 14] information to identify appropriate and graspable positions on objects. Therefore, robotic grasping applications can utilize common grippers without requiring a special gripper. However, the success of DNN models depends on the availability of a large number of representative examples of the appropriate class for training purposes. Moreover, the training process requires each sample annotated in advance, which is laborious, expensive and time-consuming. Therefore, self-supervised learning (SSL) models have become increasingly popular for robot-arm grasping applications in recent years. In SSL models, the DNN learns the required weights itself without the need for manually labelled data. In the robot-arm grasping field, SSL models commonly learn using trial-and-error methods, in which peripheral devices such as force sensors, touch sensors, or tactile sensors feed signals back to the model, and these signals are then used to annotate the grasping status as successful or not [15, 16]. However, they incur additional hardware costs for the sensors needed to provide the feedback signals and typically involve a lengthy training time. To address the problems mentioned above, we propose a two-stage grasping model in the present work. In the first stage of the proposed model, the object position is estimated using a learning-based template-matching algorithm. In detail, the learning-based template-matching algorithm is enhanced from our previous work [17]. In that work, based on the density of the high similarity scores of the measurement process, the confused similarity scores (note that confused matching scores are defined here as high similarity scores between template image and wrong targets) were detected and removed during
the matching process. In the present study, we improve the matching results by refining the estimated center coordinates of the target based on an inspection of the intensity distribution of the high similarity scores. In comparison with object detection or object tracking algorithms, template matching algorithms do not require the manual human-annotation process for a training process. In the second stage, we propose a self-rotation learning network to estimate the rotation angle of the targets. In the training process of that network, the target region (which is detected in the first stage) in the search image is cropped and self-rotated with a random angle. A Siamese network, which is constructed of two CNN-layer-based branches, is used to extract rotational representations of the cropped and rotated images. Two representations are used to calculate rotation angle by using the \( \arccos \) function. The random rotation-angle serves as a ground-truth angle for the training task. In that way, the training process of the rotation-angle estimation network does not require an expensive human-annotation process when compared with other rotation-object detection frameworks.

The main contributions of this work can be summarized as follows:

- A learning-based template-matching (LTM) algorithm is proposed to improve the position-estimation process of the matched object.
- We propose a self-rotation learning network for the rotation angle estimation to tackle the time-consuming annotation process required in traditional supervised DNN-based robot-arm grasping models.
- The experiment results on self-built datasets and a split of Cornell-grasp dataset show that the proposed model is a tradeoff between the accuracy and speed of the detection process. Moreover, the proposed model presents a good performance on unseen objects (not in the training dataset). Lastly, the practical grasping experiments show that proposed model is capable of running effectively and efficiently on a limited memory and computational resource embedded system (i.e. NVidia Jetson TX2).

The remainder of this paper is organized as follows. Section 2 briefly introduces the related work. Section 3 describes the proposed two-stage grasping model. Section 4 presents and discusses the experimental results. Finally, Sect. 5 provides some brief concluding remarks.

## 2 Related work

**Template-matching** Template-matching algorithms were widely applied in industrial manufacturing systems [18, 19] and used a variety of similarity measurement methods to evaluate the similarity between the template image and targets in the search image, including Normalization Cross-Correlation (NCC), Sum of Squared Differences (SSD), and Sum of Absolute Differences (SAD). However, when applied to rotating targets, pixel positions on rotated targets differ from those on the template image, while NCC, SSD, and SAD measurements are calculated pixel-by-pixel. Consequently, NCC, SSD, and SAD-based matching processes generally perform poorly on rotated targets [20]. In descriptor feature-based template-matching algorithms, such as Scale Invariant Feature Transform (SIFT) [21] or Oriented FAST and Rotated BRIEF (ORB) [22], key-point matching is used to measure the similarity. These algorithms were able to handle scale- and rotation-invariant objects.

In recent years, using pre-trained DNN models to extract feature maps for template-matching algorithms has been considered. There existing approaches have presented an improvement to the effective matching result. Oron et al. [23] proposed Best Buddy Similarity (BBS) method to measure the similarity between the feature maps of the template and search image, while [24, 25], and [26] were based on the diversity and the deformation amounts (DDIS), the number of co-occurrence pairs (CoTM), and quality-aware nearest neighborhood (QATM) pairs, of the pixel vectors in the feature maps for matching process. In the present work, after measuring the similarity between pixel vectors in the feature maps of the template and search image, we identify and remove confusing scores during the matching process based on the density of the high similarity scores. We then refine the estimated center coordinates for each grasping object based on the intensity distribution of the high similarity scores.

**Deep learning models** Supervised learning models have been employed to estimate potential rectangular boundary boxes of the target object [1, 2, 5, 27], to segment graspable objects in the search image [6, 28], or to measure grasping quality on depth images [4]. However, such approaches required a time-consuming and expensive annotation process. Several trial-and-error methods have been proposed for supporting the learning process of self-supervised learning models by using force signals or tactile signals as feedback information to annotate the grasping process as a success or failure. However, the training times were extremely long in both cases, i.e., 700 hours in [16] and two months in [15].

**Self-supervised learning** SSL based on the representations of two augmentations of the input image has been increasingly proposed in recent years [29, 30]. In detail, a Siamese network was used to extract representations of both augmented images, and these representations were used to measure the correlation between the two augmented images. Moreover, automatic rotating image-based self-supervised learning methods [31, 32] were recently proposed for rotation-angle classification tasks. In those approaches, the input image was rotated at a
particular angle before it was assigned to the training process. Similar to such methods, in our work, we rotate a batch image with a random angle before assigning both batch images into a Siamese network. The random rotation angle plays as the ground truth during the training process. In contrast to other rotating image-based self-supervised learning methods, we extracted representation vectors to measure the rotation angle difference between two batch images.

3 Robot-arm grasping using two-stage grasping model

Figure 1 shows the global framework of the grasping model proposed in the present work. The details of the proposed model are described in the following sections.

3.1 Translation estimation using Learning-based Template Matching (LTM) algorithm

3.1.1 Feature extraction using pre-trained mobilenet-v2

The template-matching process takes the template image $I_T$ and search image $I_S$ as the inputs and assigns them to a two networks for extracting feature maps. Both networks consist of the first convolutional layer and following four inverted residual blocks of MobileNet-v2 [33], which generate feature maps with 64 features. The MobileNet-v2 was pre-trained with the ImageNet [34] dataset. In comparison with other models, MobileNet-v2 provides a trade-off between the accuracy of the classification task and the number of model parameters and is thus particularly suitable for the implementation of deep learning models in embedded systems, which typically have relatively limited memory and computational resources. The feature maps extracted from $I_T$ and $I_S$, denoted as $T$ and $S$, respectively, are then used in the template-matching process.

3.1.2 Pairwise similarity measurement between the feature maps of the search and template images

As shown in Fig. 2, $f_{m,i}$ is referred to as the $i^{th}$ likelihood similarity map, which contains similarity scores between the $i^{th}$ pixel-vector $f_{t,i}$ in $T$ and all of the pixel vectors in $S$. The pixel vectors have a dimension of 64. The similarity score between the $i^{th}$ pixel-vector $f_{t,i}$ in $T$ and the $j^{th}$ pixel-vector $f_{s,j}$ in $S$, which is denoted as $\rho_{i,j}$, is measured using the cosine similarity measurement method:

\[ \rho_{i,j} = \frac{f_{t,i} \cdot f_{s,j}}{\|f_{t,i}\| \cdot \|f_{s,j}\|} \]
3.1.3 Finding and zeroing of confused scores in \( f_{m_i} \) based on density of the high similarity scores

For the case of a correctly identified target, the high similarity scores are expected to form one concentrated cluster in the spatial domain in the feature maps, \( S \). By contrast, for mis-matched targets, the matching scores (confused scores) are distributed more sparsely throughout \( S \). Thus, in the present work, a spatial clustering technique is used to distinguish between the matched scores and confused scores, and the confused scores are then cleared to zero in order to improve the accuracy of the matching results. As shown in Fig. 2, the process of detecting and zeroing the confused scores is implemented using a four-step procedure. The details of each step are described in the following.

**Step 1. Preprocessing scores in \( f_{m_i} \).** The similarity scores in each \( f_{m_i} \), are processed by the \texttt{softmax} function to normalize such similarity scores to range of [0,1]. Note that prior to processing, the similarity scores are divided by a temperature parameter \( \rho \) (see Eq. 2) in order to widen the gap between the low and high scores in \( f_{m_i} \) and hence emphasize the high scores [35]. Then, the high similarity scores are filtered out by comparing with the mean score of \( f_{m_i} \), \( \bar{\rho}S \). The similarity scores that are smaller than \( \bar{\rho}S \), are cleared to zeros. Then, the scores are stored in likelihood similarity maps \( f_{m_i}' \). The \texttt{softmax} function is computed as:

\[
\rho_{j}^{S}(f_{j},ft_{i}) = \frac{\exp(\rho_{j}/\tau)}{\sum_{j} \exp(\rho_{j}/\tau)}.
\]

**Step 2. Group likelihood similarity maps.** The likelihood similarity maps, \( f_{m_i}' \), are partitioned into four smaller groups of maps, \( G_r \), in accordance with the location of \( ft_{i} \) in \( T \) (see note in Step 1 of Fig. 2). In that way, in such \( f_{m_i}' \), high matched scores of \( ft_{i} \) are expected to cluster closely in spatial domain. The four groups are formulated as:

\[
G_r = \{ f_{m_i}' | \forall k \in [(r(N/4)), (r + 1)(N/4) - 1] \}.
\]
Step 3. Find maximum likelihood similarity scores in $G_r$. Across all the $fm_i$ in the same $G_r$, the maximum likelihood similarity score at each location in $fm_i$ is identified and stored in a group map, $g_r$, as follows:

$$g_r(x, y) = \max_{i \in \{y | x\} / (r+1)(k-1)} fm_i(x, y).$$

(4)

where $(x, y)$ denote coordinate of the likelihood similarity score in $fm_i$.

Step 4. Find coordinates of confused scores in $g_r$ based on density of the high similarity scores. The maximum likelihood similarity scores in each $g_r$ are clustered using the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm proposed in [36]. In particular, for each $g_r$, the coordinates of the high similarity scores, i.e., the scores with values greater than zero (while other low similarity scores in $g_r$ are cleared to zeros in Step 1), are filtered out, and the Euclidean distance method is used to measure the spatial distance between them. If two of the scores are found to have a spatial distance $(d)$ less than a certain threshold distance (denoted as eps), they are considered to be epsilon neighborhood together (members of the same neighborhood), $N_{Eps}$, as shown in Eq. (5). One high similarity score having the number of $N_{Eps}$ is equal to or greater than a threshold parameter (denoted as Minpts) be considered as the core of the spatial cluster. Moreover, the high similarity score being $N_{Eps}$ of the cluster cores are also considered as an element of that spatial cluster.

Accordingly, for correctly matched targets in the search image, it is anticipated that the high similarity scores will be densely clustered in the same region of the feature map. In other words, the value of $N_{Eps}$ is expected to be high. By contrast, the confused scores are expected to be distributed widely in the feature space, and to appear as either isolated scores, or in small randomly located clusters. In such a case, the value of $N_{Eps}$ is expected to be small. In the present work, we consider high similarity scores, which have got enough eight adjacent of high similarity scores around, as the cores of the spatial cluster. Therefore, we choose Minpts equal to eight, and eps equal to 1.5 pixels. Nevertheless, the high similarity scores that do not belong to any spatial clusters are considered as confused scores.

$$N_{Eps} = \begin{cases} 1 & d < \text{eps} \\ 0 & d \geq \text{eps} \end{cases}.$$  

(5)

Having determined the confused scores, their values are cleared to zero in order to avoid affecting the matching results. The coordinates of the confused scores are referenced to $fm$, since the original similarity measurement scores are expected to provide greater accuracy in determining the center of the target for grasping purposes than those in $fm_i$.

3.1.4 Find center coordinates of target using intensity-refinement approach based on likelihood similarity scores.

After finding and removing the confused scores in $fm$, the maximum likelihood similarity map, $fm$, between the scores in the various $fm_i$ is calculated as follows:

$$fm(x, y) = \max_{i \in [0, N-1]} fm_i(x, y).$$

(6)

It is assumed that there exists just one object in the search image correctly matched with the template image. The best-matched region, $R$, is calculated by averaging scores in $fm$ inside a region $R$ as Eq. (7). However, the target in $S$ is an unknown rotation with large height-and-width ratio. Thus, the region is with a size of $n x n$, where $n$ denotes the maximum size between height or width of $T$, such that the object can be bounded by the window irrespective of the rotation of target.

$$R^* = \arg \max_{R} \left\{ \frac{1}{n^2} \sum_{u=0}^{n} \sum_{v=0}^{n} fm(u, v) \right\}. $$

(7)

where $(u, v)$ denote coordinate of the likelihood similarity score in $fm$. The center of the best matched region is also the center of the target, which is denoted as $p(x', y')$. The score, which is referred as $p_{max}$ at the center is the confidence score of the matching process. To find the exact position of the target in the search image $Is$, $fm$ is first bilinear-interpolation resized to the size of $Is$, thus causing the coordinates $p(x', y')$ in $fm$ to be re-sized to $p(x, y)$ in $Is$. In general, re-sizing any point in a feature map up to its original size inevitably introduces position errors. Accordingly, in the present work, the re-sized target position in the search image is refined based on the intensity distribution of the likelihood similarity scores. In particular, a search is performed within a $l x l$ pixel region with $p(x, y)$ as the center of region, where $l$ denotes the ratio between the search image size and the feature map size. At each search point within this region, the average of the likelihood similarity scores is calculated within the re-sized $fm$ in a square of $m x m$, where $m$ denotes the height or width (whichever is larger) of the template image. The search position which returns the highest average value is then taken as the final position of the matching process.

 Springer
3.2 Rotation angle estimation using self-rotation learning network

3.2.1 Rotation-angle value estimation using Siamese network

As shown in Fig. 1, the rotation estimation task (i.e., the second step in the proposed grasping model) is performed using a shared-weight Siamese network consisting of two DNN-based branches. In training phase, after having estimated the center coordinates \( p(x', y') \) of the target in the feature maps of the search image (\( S \)), the cropped feature maps (\( S_r \)) is generated by cropping \( S \) at the estimated location (\( x', y' \)) with a square size of \( n \times n \) (where \( n \) is referred as mentioned above). Moreover, \( S \) is automatically rotated through a random angle (\( \theta \)) in the counter-clockwise direction with \( p(x', y') \) as the center of the rotation. Then, the rotated \( S \) is cropped at the center \( p(x', y') \) with the same size as \( S_r \) to generate the feature maps \( S_r \). The value of \( \theta \) then serves as a self-rotation label in the training phase. In real-world grasping tasks, the gripper of the robot arm is required only to rotate through a range of \([-90^\circ, 90^\circ]\) to grasp objects, which may be rotated in the range of \([0^\circ, 360^\circ]\). Therefore, the random rotation angle, \( \bar{\theta} \), is constrained to the range of \([0^\circ, 90^\circ]\). The width x height of \( S_r \) and \( S_c \) are resized to \( 32 \times 32 \) pixels so that the networks can respond with different sizes of the template images. Furthermore, to reduce the training time, the translation estimation process is first implemented for all of the images in the training dataset in order to generate a set of feature maps for the rotation estimation process. In the training process itself, \( S_c \) and \( S_r \) are fed into the Siamese rotation estimation network, and the branches output two feature descriptor vectors, \( Z_S \) and \( Z_T \). The prediction rotation angle, \( \bar{\theta} \), is then calculated as follows:

\[
\bar{\theta} = \arccos \frac{z_T \cdot \bar{z}_S}{\|z_T\| \times \|\bar{z}_S\|}.
\]

(8)

On the other hand, in inference phase, the feature map of template \( T \) is padded with an average value in \( T \) to generate a padded image \( T_p \) with a size of \( n \times n \). Then, the padded image \( T_p \) is resized to \( 32 \times 32 \) pixels before assigning to the branch 1 of the Siamese network, while the branch 2 of the Siamese network takes \( S_c \) as the input. The outputs of the branches (vectors \( Z_S \) and \( Z_T \)) are used to predict the rotation angle, \( \theta \) (see Eq. 8), initially.

3.2.2 Rotation-angle direction classification network

However, the \( \arccos \) function in Eq. (8) returns a positive angle (i.e., a counter-clockwise (\( ccw \)) rotation direction in the present work) even in the event that the real angle is negative (i.e., a clockwise direction (\( cw \))). Consequently, the feature maps of the cropped target image after the second max-pooling layer are flattened and pass through two fully connected layers to classify the true rotation sense (i.e., positive or negative) of the matched object. The network provides two classes corresponding to the two possible directions of the object (positive (\( ccw \)) or negative (\( cw \))) as the output.

4 Experimental results

The performance evaluations commenced by examining the accuracy of the proposed LTM-based translation estimation method (stage 1 of the proposed grasping model). Further experiments were then performed to evaluate the performance of the proposed grasping model. The template-matching performance was evaluated by means of simulations using the Object Tracking Benchmark 100 (OTB-100) dataset [37] on a PC equipped with an Intel Core i7-6700 CPU, 16.0 GB of memory, and an NVidia RTX 2070 GPU. Meanwhile, the grasping performance was investigated by using a self-built rotation objects dataset and a split of the Cornell grasp dataset [2]. The grasping trials were run on an NVidia Jetson TX2.

4.1 Data collection and evaluation metrics

4.1.1 Data collection

Object tracking benchmark–100 The OTB-100 dataset consists of sequences of frames. Three testing datasets were compiled (DB1, DB2, and DB3) from the frames of OTB-100, where each dataset contained 270 pairs of template images and search images. For each testing dataset, a frame \( f \) was randomly selected as the template image, and frame \((f + \Delta f)\) was selected as the search image, where \( \Delta f \) denotes the distance between the two frames in the frame sequence. The evaluations considered three different settings of \( \Delta f \) (i.e., \( 25, 50, \) and \( 100 \)) for testing datasets DB1, DB2, and DB3, respectively.

Rotation dataset The self-built rotation dataset used 22 different objects as shown in Fig. 3a). There were 8250 images collected from a fixed overhead camera with a resolution of 1280 \times 960\) pixels for the training and testing process (see Fig. 4). For the training dataset, 7000 images were split into 70\% for the training process and 30\% for validation. Each image contained one object. As shown in Fig. 3d, the images were evenly collected over the four quadrants of a circle, where the images in the first and third quadrants were taken as the positive direction class, while those in the second and fourth quadrants were taken as the negative direction class. Meanwhile, the remaining 1250 images were used to generate 2400 pairs of template images and target images for the testing dataset, in which each image contained between 1 and 4 objects, where the objects had various rotation angles and positions.
To test the grasping performance of the proposed model on unseen objects, 20 commonly used objects were additionally selected from the Cornell grasp dataset and corresponding template images were prepared (see Fig. 3b) to build an unseen grasping dataset. Each object was used to collect 100 images with different rotation angles and positions.

Mechanical-tool dataset A set of 18 mechanical-tool objects (see Fig. 3c) was used to build a grasping dataset for further testing the trial-grasping performance as unseen objects.

4.1.2 Evaluation metrics

The performance of the LTM algorithm was compared with that of several state-of-the-art methods using the area under curve (AUC) metric. With the grasping performance, the proposed grasping model was evaluated both image-wise and object-wise. With image-wise mode, the grasping performance was evaluated using the rectangle metric proposed in Jiang et al. [1], in which the object grasping task is considered to be successful if the Jaccard index (see Eq. 9) exceeds a certain threshold value and the difference between the predicted angle and the ground-truth angle lies within 30°. In the present work, the threshold value was assigned as 0.25, which is regarded as suitable for grasping tasks that do not require a high overlap between the prediction bounding box and the ground-truth bounding box [1, 2].

\[
J(g_p, g_t) = \frac{|g_p \cap g_t|}{|g_p \cup g_t|}
\] (9)

where \(J(g_p, g_t)\) denotes the Jaccard index, \(g_p\) is the predicted oriented bounding box and \(g_t\) is the ground-truth oriented bounding box, while with the object-wise mode, we identified a practical grasp as a successful grasp when the gripper of the robot arm picks a matched object successful at the position which corresponds to the center of the template image and the oriented grasping without generating collision between the gripper and the object (see the examples of the successful and failed grasping cases in Fig. 6).

4.2 Training and inference processes

Training Process The training process for rotation-angle estimation uses the mean square error (MSE) as the loss function. On the other hand, to classify the direction of objects, feature maps of \(S_c\) on the lower branch of the Siamese network are passed through two fully connected layers. This training process uses the cross-entropy as the loss function to measure the difference between the output of classification and the actual direction of objects. The final loss function of the proposed model is built as follows:

\[
\mathcal{L}(w) = \lambda_1 \mathcal{L}_1(w) + \lambda_2 \mathcal{L}_2(w)
\] (10)

where \(\mathcal{L}_1(w) = \frac{1}{B} \sum_i (\theta_i - \hat{\theta}_i)^2\) and \(\mathcal{L}_2(w) = \frac{1}{B} \sum_i (p_i \log(\hat{p}_i))^2\) are loss functions of self-rotation learning process and classification learning process, respectively, in which \(B\) denotes the batch size of training data, \(p_i\) and \(\hat{p}_i\) are predicted rotation direction and ground-truth rotation direction, respectively, while \(\lambda_1\) and \(\lambda_2\) are hyper-parameters to weigh two learning processes. Moreover, we use Adam optimizer with a learning rate of 0.0001 to optimize the model. The network was trained in 150 epochs on a desktop PC equipped with an Intel Core i7-6700 CPU and an NVidia RTX 2070 GPU.
Inference Process To reduce the processing time, the template images are extracted and generated the rotation representation in advance. In the real-world grasping process, the proposed DNN model is run on the NVidia Jetson TX2. The objects are put on a plane with an area of 40cm x 30cm. In the inference process, the location and rotation angle of objects are converted into a 3D coordinate of the robot arm based on a pre-calibrated transformation matrix between the 3D camera and 3D robot arm base coordinates. That 3D coordinate is transferred to the controller of the robot arm via TCP/IP protocol to move the end-effector to grasp objects.

4.3 Experimental results and discussions

4.3.1 Performance of LTM algorithm on OTB-100

The translation estimation performance of the proposed LTM algorithm was compared with that of four other deep feature-based template-matching algorithms (BBS [23], DDIS [24], CoTM [25], and QATM [26]). Figure 5 shows the AUC performance of the various methods when applied to the DB1, DB2, and DB3 databases with $\Delta f = 25$, $50$, and $100$, respectively. It is seen that the proposed LTM method achieves a higher AUC score than any of the other methods for all three databases. As expected, the maximum AUC score ($0.724$) is obtained for DB1 with the lowest frame separation distance of $\Delta f = 25$. The AUC value reduces slightly to $0.653$ and $0.593$ for datasets DB2 and DB3 with $\Delta f = 50$ and $100$, respectively. However, the AUC value is consistently higher than that of the other DNN-based template-matching methods. In other words, the results confirm the effectiveness of clearing the confused scores to zero and employing an intensity-based refinement step to improve the accuracy of the template-matching process.

4.3.2 Performance of the proposed grasping model on rotation dataset

The performance of the proposed DNN model was evaluated initially using the self-built rotation dataset. The grasping accuracy was compared with that of two rotation- and scale-invariant template-matching algorithms, namely SIFT [21] and ORB [22]. Both algorithms match certain key points in the template and search images and use a homography matrix based on these key points to find the oriented bounding box of the target in the search image. The performance of the three methods was evaluated both image-wise and object-wise. The former experiments involved 2400 pairs of template images and target images. The latter involved 22 objects, with each object placed with 20 different rotation angles and positions evenly divided into four cases: one-, two-, three-, and four-object on screen. The corresponding results are presented in Table 1. Note that the mean error was calculated based on results with the Jaccard index $J(g_p, g_t)$ greater than zero. It means that there is an overlap between...

Table 1 The performance comparison on Rotation dataset

| Methods          | Image-wise | Object-wise |
|------------------|------------|-------------|
|                  | Acc(%)     | PC-Time(ms) | ME of Angle(degree) | Acc(%) | TX2-Time(ms) |
| Loew [21]        | 79.8       | 356.0       | 5.9                  | 79.1   | 815.8        |
| Rublee et al. [22]| 75.8       | 78.2        | 7.6                  | 66.0   | 86.3         |
| Ours             | 92.6       | 116.6       | 5.9                  | 88.2   | 642.7        |

Acc stands for the average accuracy, PC-Time for the speed performance on PC, ME of angle for the mean error of the rotation angle, and TX2-Time for the speed performance on NVidia Jetson TX2.
the prediction bounding box and the ground-truth bounding box. Of the three methods, although ORB provides the fastest matching speed, its accuracy (75.8%) is significantly lower than that of our model (92.6%). On the other hand, our model achieves a higher object-wise accuracy (88.2%) than either ORB (66%) or SIFT (79.1%). The results from the image-wise and object-wise modes indicate that the proposed DNN model, which is based on deep features, provides a more robust detection performance than two of the most commonly used template-matching approaches in practical object detection applications. Note that SIFT, ORB, and the proposed model do not consume an expensive human-annotation process. Moreover, the mean error result of the rotation angle shows that the proposed method to estimate the rotation angle by calculating the \( \arccos \) function between two rotation representations can respond to the practical grasping task. The examples of the successful and failed grasping in the real-world grasping experiments on the Rotation dataset are shown in Fig. 6a. As shown in the first- and third columns in Fig. 6a, the objects were detected at the center of the template image and the orientations were estimated parallel with the width of the objects. Therefore, in those cases, the gripper of the robot arm was executed successful grasps. While in the second- and fourth-columns, the estimated orientations of the objects were detected with low accuracies. As a result, they generated low qualities of the grasping cases, although the gripper could pick objects. In those cases, they were defined as failed-grasping cases.

### 4.3.3 Performance of the proposed grasping model on a split of Cornell-Grasp dataset

The performance of the proposed model on Cornell-grasp objects was evaluated both image-wise and object-wise. The former experiments involved 2000 pairs of template images and target images. The latter involved 20 objects, with each object was individually tested with 10 different rotation angles and positions. Table 2 compares the unseen grasping accuracy of the proposed grasping model with that of supervised deep learning-based methods proposed in the literature and trained on the Cornell Grasp dataset. The results indicate that although the average time of the proposed model is longer than that of the method in Morrison et al. [4], the accuracy performance of the proposed model is higher than that of other compared methods in both modes. This implies that the proposed model can achieve the desired balance between grasping accuracy and processing speed. Moreover, with the Jaccard index equal to 0.5 (the overlap between the predicted object and the ground-truth is over 50%), the proposed model still obtained a good performance with 82.5% accuracy. The result shows that using the learning-based template matching in the proposed model obtained a higher effective

| Methods          | Image-wise Acc(%) | Image-wise Speed(ms) | Object-wise Acc(%) | Object-wise Speed(ms) |
|------------------|-------------------|----------------------|-------------------|----------------------|
| Jiang et al [1]  | 60.5              | 5000                 | 58.3              | 5000                 |
| Lenz et al. [2]  | 73.9              | 1350                 | 75.6              | 1350                 |
| Morrison et al. [4] | 75.8            | 19                   | 75.3              | 19                   |
| Redmon and Angelova [5] | 88.0          | 303                  | 87.1              | 303                  |
| Wang et al. [6]  | 85.3              | 141                  | -                 | -                    |
| Asif et al. [28] | 88.2              | -                    | 87.5              | -                    |
| Karaoguz and Jensfelt [27] | 88.7            | 200                  | -                 | -                    |
| **Ours (IOU ≥ 25)** | **90.5**       | **112.5**            | **88.5**          | 603**                |
| **Ours (IOU ≥ 50)** | **82.4**       | **114.1**            | -                 | -                    |

\(^{a}\) Acc and Speed stand for the average accuracy and the speed performance, respectively

\(^{b}\) The speed performance was implemented on NVidia Jetson TX2
accuracy than that of some compared grasping approaches in the location estimation task, while the proposed model does not consume an expensive human-annotation as the compared grasping approaches. The examples of the successful and failed grasping in the real-world grasping experiments on the Cornell-Grasp dataset are shown in Fig. 6b. Similar to Sect. 4.3.2, the first- and third-columns in Fig. 6b show successful-grasps with high-quality grasping cases, while the second- and fourth-columns shows low rotation-angle estimations and low-quality grasps which are identified as failed-grasping cases in our work.

4.3.4 Performance of the proposed grasping model on mechanical-tool dataset

Eighteen mechanical tools were further used as unseen objects to test the practical-grasping performance of the proposed algorithm, in which the performance was executed on NVidia Jetson TX2. Each mechanical object’s grasping performance was implemented ten times with different positions and rotation angles. Table 3 shows the grasping accuracy (82.8%) and execution speed (578ms) of the algorithm performance. It indicates that using the template-matching algorithm (does not need the training process) to estimate the location of targets and the Siamese network for estimating the rotation angle, the proposed model is capable of effective and efficient performance on unseen targets. Figure 7 shows the details of the grasping performance with the different aspect ratios of the template images. The figure shows that the success rate of the proposed model is affected by the high aspect ratio objects. When the aspect ratio increased, the success rate was reduced. Figure 8 shows examples of detection results for the grasping process on the mechanical-tool dataset. Figure 8a depicts the detection results at the center of the object, while Fig. 8b shows that the proposed model can detect specific parts of the object with the corresponding template-images. Moreover, with random geometry objects, such as tubes, pliers, and cables, in order to grasp such objects at an expected position, the template images were generated by cropping at graspable positions.

| Methods       | Acc(%) | Speed(ms) |
|---------------|--------|-----------|
| Proposed model| 82.8   | 578       |

Table 3 The performance of the proposed model on Mechanical-tool dataset

$Acc$ stands for the average accuracy

Fig. 7 The success rates of the grasping performance were executed on the mechanical-tool dataset with different aspect ratios of the template-image size

Fig. 8 The examples of the detection results on the mechanical-tool dataset. (a) shows the detection results at the center of objects; (b) shows the detection results at a specific part of an object; (c) shows the detection results on random geometry objects; and (d) shows the detection results on cables.
area of the object (see Fig. 8c, d). Therefore, the proposed model is able to detect not only objects at the center, but also areas on objects that are expected to be grasped.

5 Conclusion

This work has proposed a two-stage grasping model for robot-arm grasping applications based on a template-matching algorithm (stage 1) and self-rotation learning (SRL) network (stage 2). In the proposed model, the robustness of the position estimation process is improved by detecting and zeroing confused similarity scores in the likelihood similarity map using a spatial clustering algorithm. In the proposed grasping model, the data required for the rotation estimation training process are self-labeled by randomly rotating the input image, and a Siamese network estimates the rotation angle of the object. The experimental results have shown that the proposed LTM template-matching algorithm achieves a higher AUC score than other deep feature-based template-matching algorithms proposed in the literature, while the success rate of the proposed grasping model in practical grasping trials is significantly higher than that of other supervised grasping models. Moreover, the proposed model is capable of working effectively on untrained objects based on LTM and SRL. According to the grasping accuracy of the proposed model on real objects, this study may be applicable to classifying, grasping, and organizing industrial products inside packaging and logistics systems. However, the experimental results also show that the computational time of the LTM algorithm (due to the need to detect and remove the confused scores) and the accuracy performance on objects with high aspect ratios should be taken into consideration.

Supplementary information The online version contains supplementary material available at https://doi.org/10.1007/s00170-022-09374-y.

Author contribution All authors contributed to the study conception and design. Material preparation, data collection, analysis, and writing—original draft preparation were performed by Minh-Tri Le; supervision, project administration, writing—review and editing were performed by Jenn-Jier James Lien. All authors read and approved the final manuscript.

Funding This study was supported in part by the Ministry of Science and Technology (MOST) of Taiwan, R.O.C., under Grant No. MOST 110-2221-E-006-179. The additional support provided by Tongtai Machine & Tool Co., Ltd. (Taiwan) and Control Technology Co., Ltd. (Taiwan) is also gratefully acknowledged.

Data availability Not applicable.

Code availability Not applicable.

Declarations

Ethics approval The authors state that the present work is in compliance with the ethical standards.

Consent to participate There is no consent to participate needed in the present study.

Consent for publication There is no consent to publish needed in the present study.

Conflicts of interest The authors declare no competing interests.

References

1. Jiang Y, Moseson S, Saxena A (2011) Efficient grasping from rgbd images: Learning using a new rectangle representation. In: IEEE Int Conf Robot Autom (ICRA), pp 3304–3311
2. Lenz I, Lee H, Saxena A (2015) Deep learning for detecting robotic grasps. Int J Robot Res 34(4–5):705–724
3. Li CHG, Chang YM (2019) Automated visual positioning and precision placement of a workpiece using deep learning. Int J Adv Manuf Technol 104(9):4527–4538
4. Morrison D, Corke P, Leiner J (2020) Learning robust, real-time, reactive robotic grasping. Int J Robot Res 39(2–3):183–201
5. Redmon J, Angelova A (2015) Real-time grasp detection using convolutional neural networks. In: 2015 IEEE Int Conf Robot Autom (ICRA), IEEE, pp 1316–1322
6. Wang Z, Li Z, Wang B, Liu H (2016) Robot grasp detection using multimodal deep convolutional neural networks. Adv Mech Eng 8(9):1687814016668077
7. Zhao D, Sun F, Wang Z, Zhou Q (2021) A novel accurate positioning method for object pose estimation in robotic manipulation based on vision and tactile sensors. Int J Adv Manuf Technol 116(9):2999–3010
8. Elangovan N, Gerez L, Gao G, Liarokapis M (2021) Improving robotic manipulation without sacrificing grasping efficiency: a multi-modal, adaptive gripper with reconfigurable finger bases. IEEE Access 9:83298–83308
9. Michalos G, Dimoulas K, Mparis K, Karagiannis P, Makris S (2018) A novel pneumatic gripper for in-hand manipulation and feeding of lightweight complex parts—a consumer goods case study. Int J Adv Manuf Technol 97(9):3735–3750
10. Spiliotopoulos J, Michalos G, Makris S (2018) A reconfigurable gripper for dexterous manipulation in flexible assembly. Inventions 3(1):4
11. Kocic M, Stork JA, Haustein JA, Kragic D (2017) Affordance detection for task-specific grasping using deep learning. In: 2017 IEEE-RAS 17th Int Conf Humanoids, IEEE, pp 91–98
12. Rezapour Lakani S, Rodriguez-Sánchez AJ, Piater J (2019) Towards affordance detection for robot manipulation using affordance for parts and parts for affordance. Auton Robots 43(5):1155–1172
13. Mahler J, Matl M, Liu X, Li A, Gealy D, Goldberg K (2018) Dext-net 3D: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning. In: 2018 IEEE Int Conf Robot Autom (ICRA), IEEE, pp 5620–5627
14. Monica R, Aleotti J (2020) Point cloud projective analysis for part-based grasp planning. IEEE Robot Autom Lett 5(3):4695–4702
15. Levine S, Pastor P, Krizhevsky A, Ibarz J, Quillen D (2018) Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. Int J Robot Res 37(4–5):421–436
16. Pinto L, Gupta A (2016) Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In: 2016 IEEE Int Conf Robot Autom (ICRA), IEEE, pp 3406–3413

17. Le MT, Lien JJJ (2021) Learning-based template matching for robot arm grasping. In: 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, pp 1763–1768

18. Chen F, Ye X, Yin S, Ye Q, Huang S, Tang Q (2019) Automated vision positioning system for dicing semiconductor chips using improved template matching method. Int J Adv Manuf Technol 100(9):2669–2678

19. Zhong F, He S, Li B (2017) Blob analyzation-based template matching algorithm for led chip localization. Int J Adv Manuf Technol 93(1):55–63

20. Desai BK, Pandya M, Potdar M (2013) Comparison of various template matching techniques for face recognition. Int J Eng Res Dev 8(10):16–18

21. Loew DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vision

22. Rublee E, Rabaud V, Konolige K, Bradski G (2011) Orb: an efficient alternative to sift or surf. In: IEEE Int Conf Comput Vision, pp 2564–2571

23. Oron S, Dekel T, Xue T, Freeman WT, Avidan S (2017) Best-buddies similarity–robust template matching using mutual nearest neighbors. IEEE Trans Pattern Anal Mach Intell 40(8):1799–1813

24. Talmi I, Mechrez R, Zelnik-Manor L (2017) Template matching with deformable diversity similarity. In: Proc Conf Comput Vis Pattern Recognit, pp 1751–1759

25. Kat R, Jevnisek R, Avidan S (2018) Matching pixels using co-occurrence statistics. In: Proc Conf Comput Vis Pattern Recognit, pp 175–183

26. Cheng J, Wu Y, AbdAlmageed W, Natarajan P (2019) QATM: Quality-aware template matching for deep learning. In: Proc Conf Comput Vis Pattern Recognit, pp 11553–11562

27. Karaoguz H, Jensfelt P (2019) Object detection approach for robot grasp detection. In: 2019 IEEE Int Conf Robot Autom (ICRA), IEEE, pp 4953–4959

28. Asif U, Tang J, Harrer S (2018) Graspnet: an efficient convolutional neural network for real-time grasp detection for low-powered devices. In: IJCAI, pp 4875–4882

29. Chen T, Kornblith S, Norouzi M, Hinton G (2020) A simple framework for contrastive learning of visual representations. In: International Conference on Machine Learning, PMLR, pp 1597–1607

30. Grill JB, Strub F, Altché F, Tallec C, Richemond P, Buchatskaya E, Doersch C, Avila Pires B, Guo Z, Gheshlaghi Azar M et al (2020) Bootstrap your own latent—a new approach to self-supervised learning. Adv Neural Info Process Syst 33:21271–21284

31. Feng Z, Xu C, Tao D (2019) Self-supervised representation learning by rotation feature decoupling. In: Proc Conf Comput Vis Pattern Recognit, pp 10364–10374

32. Li X, Hu X, Qi X, Yu L, Zhao W, Heng PA, Xing L (2021) Rotation-oriented collaborative self-supervised learning for retinal disease diagnosis. IEEE Trans Med Imag

33. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC (2018) Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proc Conf Comput Vis Pattern Recognit, pp 4510–4520

34. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: A large-scale hierarchical image database. In: IEEE Conf Comput Vis Pattern Recognit, IEEE, pp 248–255

35. Wu Z, Xiong Y, Yu SX, Lin D (2018) Unsupervised feature learning via non-parametric instance discrimination. In: Proc Conf Comput Vis Pattern Recognit, pp 3733–3742

36. Ester M, Kriegel HP, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. AAAI Press, pp 226–231

37. Wu Y, Lim J, Yang MH (2013) Online object tracking: a benchmark. In: Proc Conf Comput Vis Pattern Recognit, pp 2411–2418

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.