Research Square

Preprints are preliminary reports that have not undergone peer review. They should not be considered conclusive, used to inform clinical practice, or referenced by the media as validated information.

Robot Arm Grasping Using Learning-Based Template Matching and Self-Rotation Learning Network

Minh-Tri Le (lmt.ncku@gmail.com)
National Cheng Kung University https://orcid.org/0000-0002-0673-8757
Jenn-Jier James Lien

Research Article

Keywords: Self-supervised learning, deep learning, robot arm grasping, template matching

Posted Date: March 8th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1402918/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Manuscript

Robot Arm Grasping Using Learning-based Template Matching and Self-Rotation Learning Network

Minh-Tri Le[1] and Jenn-Jier James Lien[1*]

[1*]Department of Computer Science and Information Engineering, National Cheng Kung Uninversity, No.1, University Road, Tainan, 701, Taiwan.

*Corresponding author(s). E-mail(s): jjlien@csie.ncku.edu.tw;
Contributing authors: lmtkdt@gmail.com;

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 Jiang et al (2011); Lenz et al (2015); Redmon and Angelova (2015); Wang et al (2016, 2020); Zhao et al (2021); Li and Chang (2019); Morrison et al (2020). 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. 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 Pinto and Gupta (2016);

Levine et al (2018). 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 Le and Lien (2021). 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 to object detection or object tracking algorithms, template matching algorithms does 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 *arccosine* 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, Section 5 provides some brief concluding remarks.

2 Related Work

Template-matching. Template-matching algorithms are widely applied in industrial manufacturing systems Chen et al (2019); Zhong et al (2017); Wang et al (2018); Annaby et al (2019); Tsai and Huang (2018) and use 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, such methods generally achieve a poor matching performance on rotated objects. Descriptor feature-based template-matching algorithms, such as Scale Invariant Feature Transform (SIFT) LOEW (2004) or Oriented FAST and Rotated BRIEF (ORB) Rublee et al (2011), used matching points to measure the similarity and are able to deal with invariant scale and rotation objects.

In recent years, using pre-trained DNN models to extract feature maps for template-matching algorithms was considered. There existed approaches presented to improve the effective matching result. Oron et al. Oron et al (2017) proposed a method to find best-buddy pairs in the feature maps of the template and search image. Then using a similarity measurement method called Best Buddy Similarity (BBS) for measuring the similarity process. Moreover, authors in Talmi et al (2017); Kat et al (2018) proposed methods to identify nearest neighborhood (NN) pairs between the pixel vectors in the feature maps of the template and search images for template-matching processes based on the diversity and the deformation amounts (DDIS), or based on the number of co-occurrence pairs (CoTM), of the pixel vectors. While Cheng et al. Cheng et al (2019) presented a method to identify quality-aware NN pairs for the template matching process (QATM) by measuring similarity between pairs of pixel vectors in the deep features of the template image and search image using a pair-wise similarity approach. Accordingly, in the present work, the template-matching algorithm employed in the first stage of the proposed DNN model used a pre-trained DNN model to extract feature maps of template and search images for measuring the similarity process. After that, based on the density of the high similarity scores to identify and remove confused scores during the matching process, and then based on the intensity distribution of the high similarity scores to refine the estimated center coordinates of the grasping objects.

Deep learning models for robot-arm grasping. Supervised learning models have been employed to estimate potential rectangular boundary boxes of the target object Jiang et al (2011); Lenz et al (2015); Redmon and Angelova (2015); Karaoguz and Jensfelt (2019), to segment graspable objects in the search image Wang et al (2016); Asif et al (2018), or to measure grasping quality on depth images Morrison et al (2020). However, such approaches required a time-consuming and expensive annotation process. The application of SSL models to the robot-arm grasping problem has attracted great interest in the recent literature. Several trial-and-error methods have been proposed for supporting the learning process of SSL models by using force signals or tactile signals as feedback information to annotate the grasping process as a success or failure Pinto and Gupta (2016); Levine et al (2018). However, the training times were extremely long in both cases, i.e., 700 hours in Pinto and Gupta (2016) and two months in Levine et al (2018).

Self-supervised learning model for rotation-angle estimation. The input images are self-rotated by multiples of 90 degrees and assigned into a DNN model to extract rotation representation features for predicting the rotation angle of the input image Feng et al (2019); Li et al (2021). In approach Gidaris et al (2018), a transformation set was generated to transform the input images with different rotation angles. The transformed images were assigned into an unsupervised DNN model to train the model to classify the transformation of images. Such approaches predicted discrete angles with a rotation-angle interval of 90-degree, so the accuracy of the rotation-angle estimation process was with discrete angles. In our work, the rotation angle is identified using *arccosine* operator between rotation representations of the template image and the target candidate for estimating the rotation-angle of object. Therefore, the rotation-angle estimation result can obtain a smaller rotation-angle interval.

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 Sandler et al (2018), which generate feature maps with 64 features. The MobileNet-v2 was pre-trained with the ImageNet Deng et al (2009) 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

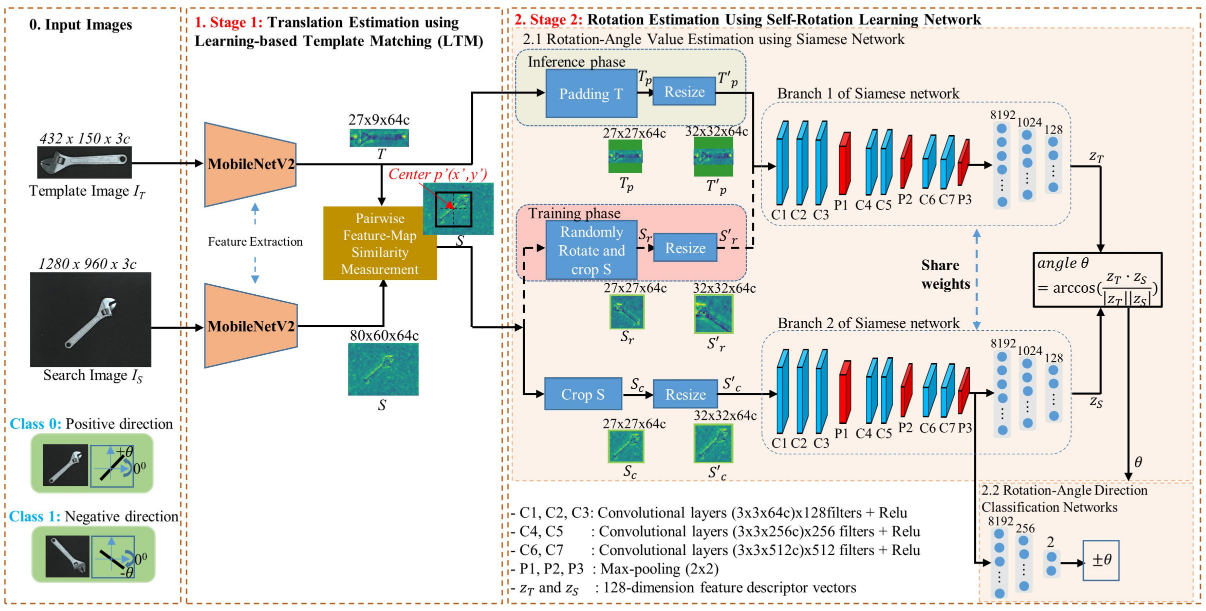

0. Input Images
432 x 150 x 3c
Template Image I_T
1280 x 960 x 3c
Search Image I_S
Class 0: Positive direction
Class 1: Negative direction
1. Stage 1: Translation Estimation using Learning-based Template Matching (LTM)
MobileNetV2
Feature Extraction
MobileNetV2
27x9x64c
T
Center p'(x',y')
Pairwise Feature-Map Similarity Measurement
S
80x60x64c
S
2. Stage 2: Rotation Estimation Using Self-Rotation Learning Network
2.1 Rotation-Angle Value Estimation using Siamese Network
Inference phase
Padding T
Resize
Training phase
Randomly Rotate and crop S
Resize
Crop S
Resize
Branch 1 of Siamese network
Branch 2 of Siamese network
Share weights
C1 C2 C3 P1 C4 C5 P2 C6 C7 P3
8192 1024 128
angle θ = arccos($\frac{z_T \cdot z_S}{|z_T||z_S|}$)
2.2 Rotation-Angle Direction Classification Networks
8192 256 2
±θ
- C1, C2, C3: Convolutional layers (3x3x64c)x128filters + Relu
- C4, C5 : Convolutional layers (3x3x256c)x256 filters + Relu
- C6, C7 : Convolutional layers (3x3x512c)x512 filters + Relu
- P1, P2, P3 : Max-pooling (2x2)
- z_T and z_S : 128-dimension feature descriptor vectors

Fig. 1: The global framework of the proposed grasping model.

template-matching process, as described in the following.

3.1.2 Pairwise Similarity Measurement between the Feature Maps of the Search and Template Images

As shown in Fig. 2, fm_i is referred to as the i^{th} likelihood similarity map, which contains similarity scores between the i^{th} pixel-vector ft_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 ft_i in T and the j^{th} pixel-vector fs_j in S, which is denoted as $\rho^{\mathrm{i}}_{\mathrm{j}}$, is measured using the cosine similarity measurement method:

$$\rho^{\mathrm{i}}_{\mathrm{j}}(fs_j, ft_i) = \frac{ft_i \cdot fs_j}{\|ft_i\| \times \|fs_j\|}. \tag{1}$$

$$i = 0 \rightarrow N;\ j = 0 \rightarrow M$$

where w_T and h_T are the width and height of the feature maps in T; while w_S and h_S are the width and height of the feature maps in S, respectively. $N = w_T$ x h_T denotes the totally number of pixel-vector in T, while $M = w_S$ x h_S denotes the totally number of pixel-vector in S. In an ideal matching process, one ft_i matches correctly with just one fs_j, where this pixel vector belongs to the feature maps region corresponding to the target. In such a situation, the similarity score is referred to as a “matched score”. However, in some cases, ft_i may achieve high similarity scores with multiple pixel vectors fs_j located in regions of the feature maps not associated with the target. In such a case, the similarity score is said to be a “confused score”. The presence of these confused scores can seriously degrade the accuracy of the matching results, and thus they should be eliminated before the center coordinates of the target object are evaluated.

3.1.3 Finding and Zeroing of Confused Scores in fm_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 fm_i. The similarity scores in each fm_i are processed by the $softmax$ 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 ρ (see Eq. 2) in order to widen the gap between the low and high scores in fm_i and hence emphasize the high scores Wu et al (2018). Then, the high similarity scores are filtered out by comparing with the mean score of fm_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 fm'_i. The $softmax$ function is computed as:

$$\rho S^{\mathrm{i}}_{\mathrm{j}}(fs_j, ft_i) = \frac{exp(\rho^{\mathrm{i}}_{\mathrm{j}}/\tau)}{\sum\limits_{j}^{M} exp(\rho^{\mathrm{i}}_{\mathrm{j}}/\tau)}. \tag{2}$$

Step 2. Group likelihood similarity maps. The likelihood similarity maps, fm'_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 fm'_i, high matched scores of ft_i are expected to cluster closely in spatial domain. The four groups are formulated as,

$$G_r = \{fm'_k | \forall k \in [(r(N/4)), (r+1)(N/4) - 1]\}. \tag{3}$$

where r denotes the group number, i.e., $i = 0 \to 3$; k is fm'_i number in each group G_r.

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:

$$g_r(x, y) = \max_{i \in [(rk),(r+1)k-1]} fm'_i(x, y). \tag{4}$$

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 Ester et al (1996). 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} are 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.

$$N_{Eps} = \begin{cases} 1 & d < eps \\ 0 & d \geq eps \end{cases}. \tag{5}$$

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_i since the original similarity measurement scores are expected to

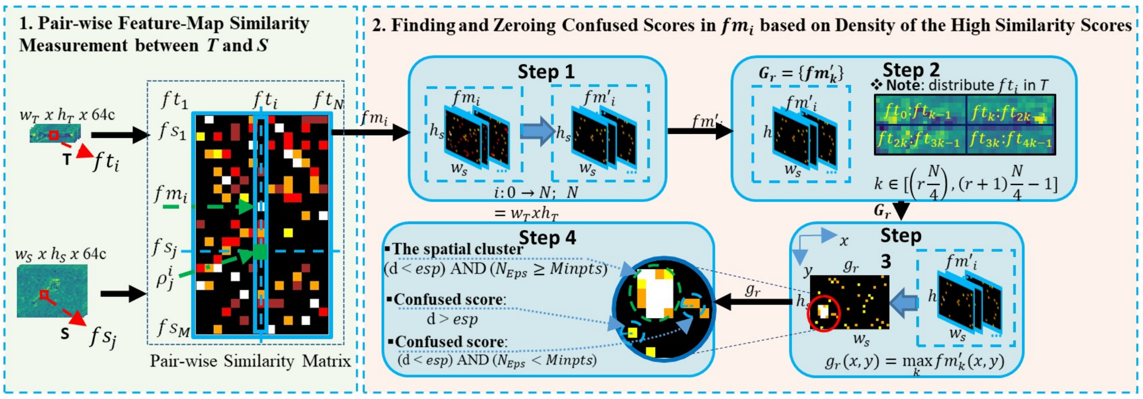

1. Pair-wise Feature-Map Similarity Measurement between T and S
2. Finding and Zeroing Confused Scores in fm_i based on Density of the High Similarity Scores
Pair-wise Similarity Matrix
Step 1
Step 2
Note: distribute ft_i in T
Step 3
Step 4
The spatial cluster (d < esp) AND ($N_{Eps} \geq Minpts$)
Confused score: d > esp
Confused score: (d < esp) AND (N_{Eps} < Minpts)
$i: 0 \to N$; $N = w_T x h_T$
$k \in [(r\frac{N}{4}), (r+1)\frac{N}{4} - 1]$
$g_r(x,y) = \max_k fm'_k(x,y)$
$G_r = \{fm'_k\}$

Fig. 2: Finding and zeroing confused scores based on density of the high similarity scores.

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_i, the maximum likelihood similarity map, fm, between the scores in the various fm_i is calculated as follows:

$$fm(x,y) = \max_{i=[0,N-1]} fm_i(x,y). \qquad (6)$$

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.

$$R^* = \operatorname*{argmax}_R \{\frac{1}{n^2}\sum_{u=0}^{n}\sum_{v=0}^{n} fm(u,v)\}. \qquad (7)$$

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 ρ_{max}, at the center is the confidence score of the matching process. To find the exact position of the target in the search image I_s, fm is first bilinear-interpolation re-sized to the size of I_s, thus causing the coordinates $p(x',y')$ in fm to be re-sized to $p(x,y)$ in I_s. 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.

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_c) is generated by cropping S at the estimated location (x', y') with a square size of nxn (where n is referred as mentioned above). Moreover, S is automatically rotated through a random angle ($\hat{\theta}$) 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_c to generate the feature maps S_r. The value of $\hat{\theta}$ 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 [$-90°$, $90°$] to grasp objects, which may be rotated in the range of [0°, 360°]. Therefore, the random rotation angle, $\hat{\theta}$, is constrained to the range of [0°, 90°]. The width x height of S_c and S_r are resized to $32x32$ 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, θ, is then calculated as follows:

$$\theta = \arccos \frac{z_T \cdot z_S}{\|z_T\| \times \|z_S\|}. \tag{8}$$

On the other hand, in inference phase, the feature maps of template T is padded with an average value in T to generate a padded image T_p with a size of n x n. Then, the padded image T_p is resized to 32 x 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, θ (see Eq. 8), initially.

3.2.2 Rotation-Angle Direction Classification Network

However, the *arccosine* 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 Wu et al (2013) 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 Lenz et al (2015). 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 Δf denotes the distance between the two frames in the frame sequence. The evaluations considered three different settings of Δf (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 with large height-to-width ratios to generate 22 template images with sizes of 220∼420 x 130∼220 pixels (see Fig. 3a). There were 8250 images collected from a fixed overhead camera with a solution of 1280 x 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.

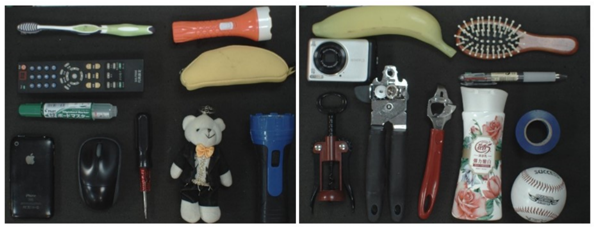
A Split of Cornell-Grasp Dataset. To test the grasping performance of the proposed model on unseen objects, 20 commonly used objects were additionally selected from the Cornell grasp dataset and corresponding template images were prepared (see Fig. 3b) to build an unseen grasping dataset. Each object was used to collect 100 images with different rotation angles and positions.

Mechanical-Tool Dataset. A set of 10 mechanical-tool objects (see Fig. 3c) was used to build a grasping dataset for further testing the trial-grasping performance on unseen objects.

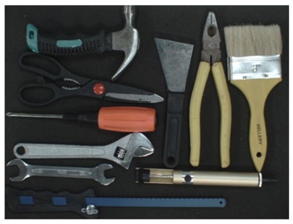
4.1.2 Evaluation Metrics

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 (using the rectangle metric to evaluate the grasping performance in the testing image) and object-wise (real-world grasping tasks). With image-wise mode, the grasping performance was evaluated using the rectangle metric proposed in Jiang et al (2011), 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 30^0. 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

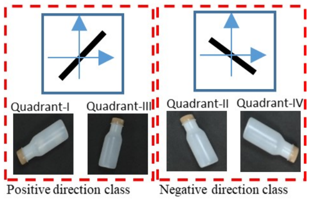
(a)



(b)



(c)



(d)

Fig. 3: The illustration of the objects used for the self-built rotation dataset in (a), the unseen dataset in (b), the mechanical-tool dataset in (c), and the direction classes for training process in (d).

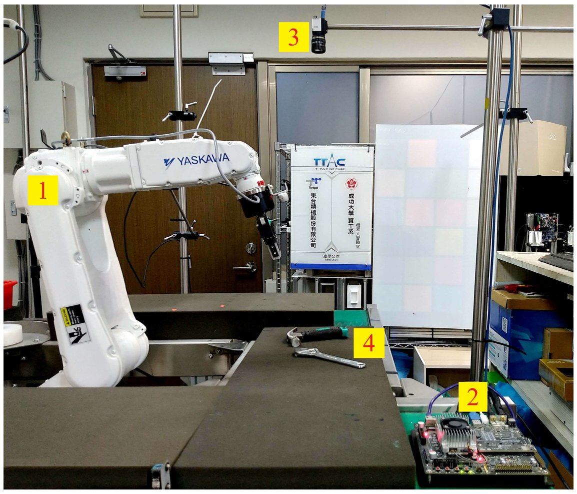


Fig. 4: The robot arm system with a Yaskawa robot arm (1), an NVidia Jetson TX2 developer kit in (2), a fixed overhead eye-to-hand camera in (3), and the illustration objects in (4).

bounding box Jiang et al (2011); Lenz et al (2015).

$$J(g_p, g_t) = \frac{|g_p \cap g_t|}{|g_p \cup g_t|}. \tag{9}$$

where $J()$ denotes as 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 with the template image 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:

$$\mathcal{L}(w) = \lambda_1 \mathcal{L}_1(w) + \lambda_2 \mathcal{L}_2(w) \tag{10}$$

where $\mathcal{L}_1(w) = \frac{1}{B}\sum_i^B (\theta_i - \hat{\theta}_i)^2$ and $\mathcal{L}_2(w) = \frac{1}{B}\sum_i^B (p_i \log(\hat{p_i}))^2$ 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 λ_1 and λ_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 and required almost a day to complete 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

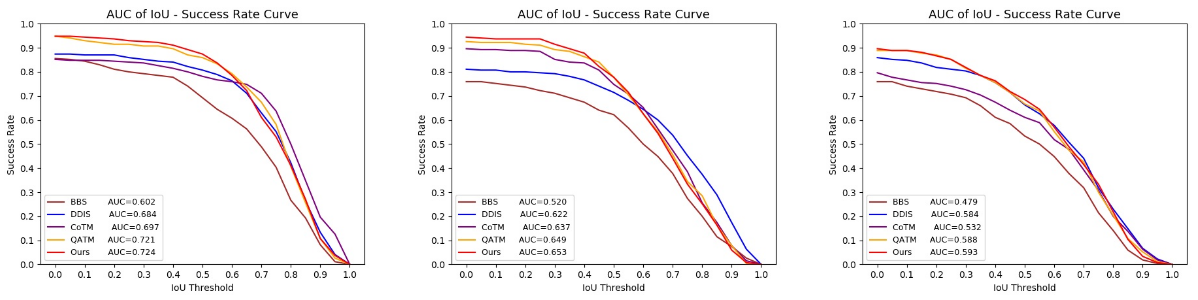
The translation estimation performance of the proposed LTM algorithm was compared with that of four other deep feature-based template-matching algorithms (BBSOron et al (2017), DDISTalmi et al (2017), CoTMKat et al (2018), and QATMCheng et al (2019)). Figure 5 shows the AUC performance of the various methods when applied to the DB1, DB2, and DB3 databases with Δf= 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 LOEW (2004) and ORB Rublee et al (2011). 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 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 *arccosine* functions 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. In object-wise mode, the camera solution was set with a resolution of 1280x960 pixels. 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. In particular, the proposed model obtains a grasping accuracy of 90.5% in the image-wise mode on PC and 88.5% in the object-wise mode on NVidia Jetson TX2. Such accuracies are higher than that of other compared methods in both modes. While although the average time of the proposed model is longer than that of the method in Morrison et al (2020), but is significantly faster than that of the remaining



(a) (b) (c)

Fig. 5: Accuracy of LTM and other template-matching algorithms for three split datasets (DB1, DB2 and DB3) from OTB-100 dataset. (a) DB1, (b) DB2, and (c) DB3. Note that the AUC values are shown in the legend. Better viewed in color

Table 1: The performance comparison on Rotation dataset

	Image-wise			Object-wise	
Methods	Acc(%)	PC-Time(ms)	ME of Angle($degree$)	Acc(%)	TX2-Time(ms)
SIFT LOEW (2004)	79.8	356.0	5.9	79.1	815.8
ORB Rublee et al (2011)	75.8	**78.2**	7.6	66.0	**86.3**
Ours	**92.6**	116.6	**5.9**	**88.2**	642.7

Note: *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.

compared methods. It indicates that the proposed model can obtain the trade-off between the grasping accuracy and the processing speed. The results presented in Table 2 also show that, although running on an embedded platform, the accuracy and speed performances of the proposed model is comparable with that of other grasping literature. 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 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 Section 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

Ten 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 (81%) and execution speed (667ms) of the algorithm performance. The accuracy performance is similar to the results in Section 4.3.3. It

Table 2: The performance comparison on a Split of Cornell-Grasp dataset

	Image-wise		Object-wise	
Methods	Acc[1](%)	Speed[1](ms)	Acc(%)	Speed(ms)
Jiang et al. Jiang et al (2011)	60.5	5000	58.3	5000
Lenz et al. Lenz et al (2015)	73.9	1350	75.6	1350
Morrison et al. Morrison et al (2020)	75.8	**19**	75.3	**19**
Redmon et al. Redmon and Angelova (2015)	88.0	303	87.1	303
Wang et al. Wang et al (2016)	85.3	141	-	-
Asif et al. Asif et al (2018)	88.2	-	87.5	-
Karaoguz et al. Karaoguz and Jensfelt (2019)	88.7	200	-	-
Ours (IOU$\geq$ 25)	**90.5**	112.5	**88.5**	603[2]
Ours (IOU$\geq$ 50)	82.4	114.1	-	-

[1]Note: *Acc* and *Speed* stand for the average accuracy and the speed performance, respectively.
[2]The speed performance was implemented on NVidia Jetson TX2.

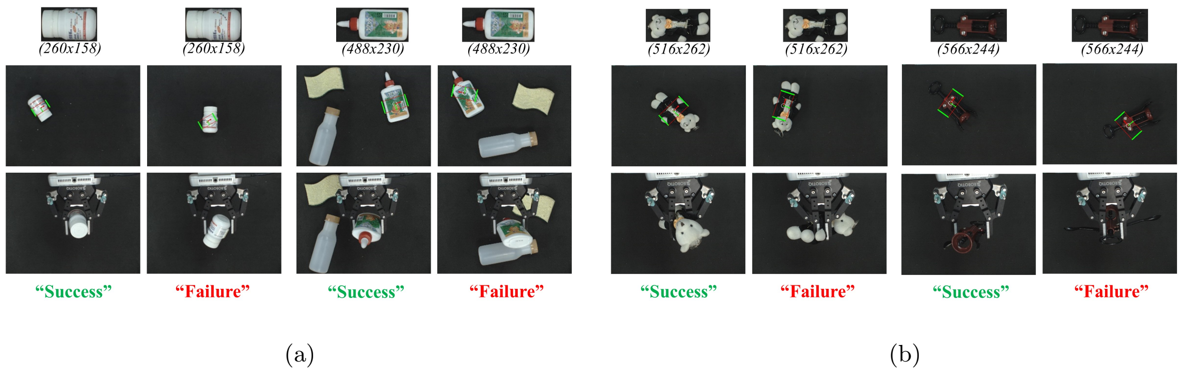


Fig. 6: The illustration of the successful and failed grasping cases on the rotation dataset (a) and the unseen dataset in the real-world grasping experiments. The first row shows the template images with their size. The second row and the third row show the examples of the detected objects and the gripper states, respectively.

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. While Fig. 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 success- and failure-grasps on the mechanical-tool dataset. It shows that with high aspect ratio objects, the template-matching process obtained a low accuracy when detecting the center of the target in failed-grasping cases with high aspect ratio objects. As a result, the orientation of the target was affected. Therefore, it affects the grasping-performance accuracy of the proposed algorithm.

5 Conclusion

This work has proposed a two-stage grasping model for robot-arm grasping applications based

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

Methods	Acc(%)	Speed(ms)
Proposed model	81.0	667

Note: *Acc* stands for the average accuracy.

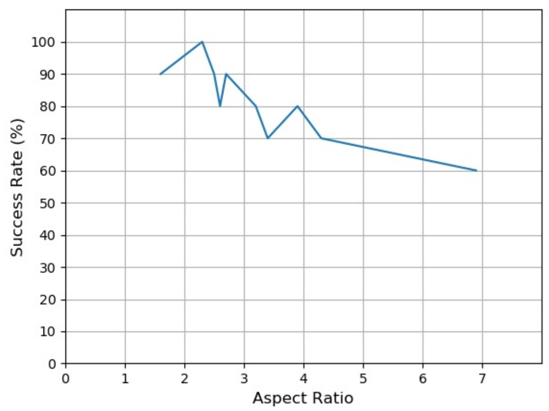


Fig. 7: The success rates of the grasping performance were executed on the mechanical-tool dataset with different aspect ratios of the template-image size.

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. The accuracy of the matching results (i.e., the center coordinate of the target object in the search image) is then further improved through a refinement process based on an inspection of the intensity distribution of the likelihood similarity scores. In the proposed grasping model, the data required for the rotation estimation training process is 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

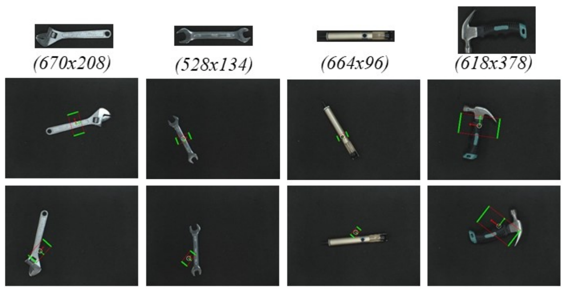


Fig. 8: The examples of the successful- and failed-grasping cases on the mechanical-tool dataset. The first row shows the template images with their size. The second and third rows show the successful- and failed-grasping images, respectively.

grasping models. Moreover, based on LTM and SRL, the proposed model is capable of effective working on untrained objects. It is noted that the proposed LTM algorithm is obtained at the expense of a longer computation time due to the need to detect and remove the confused scores and perform the intensity-based refinement process. Overall, however, the results indicate that the proposed grasping model provides an accurate and robust approach for dealing with real-world grasping tasks without the need for a time-consuming manual annotation process.

Declarations

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 Contrel Technology Co., Ltd. (Taiwan) is also gratefully acknowledged.

Conflicts of interest. The authors declare no competing interests.

Availability of data and material. Not applicable

Code availability. Not applicable

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.

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.

References

Annaby M, Fouda Y, Rushdi M (2019) Improved normalized cross-correlation for defect detection in printed-circuit boards. IEEE Transactions on Semiconductor Manufacturing 32(2):199–211

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

Chen F, Ye X, Yin S, et al (2019) Automated vision positioning system for dicing semiconductor chips using improved template matching method. The International Journal of Advanced Manufacturing Technology 100(9):2669–2678

Cheng J, Wu Y, AbdAlmageed W, et al (2019) Qatm: Quality-aware template matching for deep learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 11,553–11,562

Deng J, Dong W, Socher R, et al (2009) Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition, Ieee, pp 248–255

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

Feng Z, Xu C, Tao D (2019) Self-supervised representation learning by rotation feature decoupling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 10,364–10,374

Gidaris S, Singh P, Komodakis N (2018) Unsupervised representation learning by predicting image rotations. arXiv preprint arXiv:180307728

Jiang Y, Moseson S, Saxena A (2011) Efficient grasping from rgbd images: Learning using a new rectangle representation. In: 2011 IEEE International conference on robotics and automation, IEEE, pp 3304–3311

Karaoguz H, Jensfelt P (2019) Object detection approach for robot grasp detection. In: 2019 International Conference on Robotics and Automation (ICRA), IEEE, pp 4953–4959

Kat R, Jevnisek R, Avidan S (2018) Matching pixels using co-occurrence statistics. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 1751–1759

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

Lenz I, Lee H, Saxena A (2015) Deep learning for detecting robotic grasps. The International Journal of Robotics Research 34(4-5):705–724

Levine S, Pastor P, Krizhevsky A, et al (2018) Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. The International Journal of Robotics Research 37(4-5):421–436

Li CHG, Chang YM (2019) Automated visual positioning and precision placement of a workpiece using deep learning. The International Journal of Advanced Manufacturing Technology 104(9):4527–4538

Li X, Hu X, Qi X, et al (2021) Rotation-oriented collaborative self-supervised learning for retinal disease diagnosis. IEEE Transactions on Medical Imaging

LOEW DG (2004) Distinctive image features from scale-invariant keypoints. International journal of computer vision

Morrison D, Corke P, Leitner J (2020) Learning robust, real-time, reactive robotic grasping. The International journal of robotics research 39(2-3):183–201

Oron S, Dekel T, Xue T, et al (2017) Best-buddies similarity—robust template matching using mutual nearest neighbors. IEEE transactions on pattern analysis and machine intelligence 40(8):1799–1813

Pinto L, Gupta A (2016) Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In: 2016 IEEE international conference on robotics and automation (ICRA), IEEE, pp 3406–3413

Redmon J, Angelova A (2015) Real-time grasp detection using convolutional neural networks. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp 1316–1322

Rublee E, Rabaud V, Konolige K, et al (2011) Orb: An efficient alternative to sift or surf. In: 2011 International conference on computer vision, Ieee, pp 2564–2571

Sandler M, Howard A, Zhu M, et al (2018) Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4510–4520

Talmi I, Mechrez R, Zelnik-Manor L (2017) Template matching with deformable diversity similarity. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 175–183

Tsai DM, Huang CK (2018) Defect detection in electronic surfaces using template-based fourier image reconstruction. IEEE Transactions on Components, Packaging and Manufacturing Technology 9(1):163–172

Wang W, Wang Q, Yamane S, et al (2018) Tracking using pattern matching of keyhole in visual robotic plasma welding. The International Journal of Advanced Manufacturing Technology 98(5):2127–2136

Wang Z, Li Z, Wang B, et al (2016) Robot grasp detection using multimodal deep convolutional neural networks. Advances in Mechanical Engineering 8(9):1687814016668,077

Wang Z, Xu Y, He Q, et al (2020) Grasping pose estimation for scara robot based on deep learning of point cloud. The International Journal of Advanced Manufacturing Technology 108(4):1217–1231

Wu Y, Lim J, Yang MH (2013) Online object tracking: A benchmark. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2411–2418

Wu Z, Xiong Y, Yu SX, et al (2018) Unsupervised feature learning via non-parametric instance discrimination. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3733–3742

Zhao D, Sun F, Wang Z, et al (2021) A novel accurate positioning method for object pose estimation in robotic manipulation based on vision and tactile sensors. The International Journal of Advanced Manufacturing Technology 116(9):2999–3010

Zhong F, He S, Li B (2017) Blob analyzation-based template matching algorithm for led chip localization. The International Journal of Advanced Manufacturing Technology 93(1):55–63

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- 00ImageWiseGrasping2400ORB.xlsx
- 00ImageWiseGrasping2400SIFT.xlsx
- 00ImageWiseGrasping2400SelfLearn.xlsx
- CompareGraspingResultsonRotationData.xlsx
- ImageWiseCornellObject.xlsx
- ORBGraspingRotationDataset1Object.xlsx
- ORBGraspingRotationDataset2Objects.xlsx
- ORBGraspingRotationDataset3Objects.xlsx
- ORBGraspingRotationDataset4Objects.xlsx
- OTBDelta100IoUResults.xlsx
- OTBDelta25IoUResults.xlsx
- OTBDelta50IoUResults.xlsx
- ObjectWiseCornellObjects.xlsx
- ObjectWiseMechanicalToolObjects.xlsx
- ProposedMethodPracticalGrasp1Object.xlsx
- ProposedMethodPracticalGrasp2Objects.xlsx
- ProposedMethodPracticalGrasp3Objects.xlsx
- ProposedMethodPracticalGrasp4Object.xlsx
- SIFTGraspingRotationDataset1Object.xlsx
- SIFTGraspingRotationDataset2Objects.xlsx
- SIFTGraspingRotationDataset3Objects.xlsx
- SIFTGraspingRotationDataset4Objects.xlsx
- SelfRotationLearningGraspingDemo.mp4