Cognitive Computation
https://doi.org/10.1007/s12559-021-09966-y

CSA6D: Channel-Spatial Attention Networks for 6D Object Pose Estimation

Tao Chen[1] · **Dongbing Gu**[1]

Received: 13 October 2021 / Accepted: 20 November 2021

Abstract
6D object pose estimation plays a crucial role in robotic manipulation and grasping tasks. The aim to estimate the 6D object pose from RGB or RGB-D images is to detect objects and estimate their orientations and translations relative to the given canonical models. RGB-D cameras provide two sensory modalities: RGB and depth images, which could benefit the estimation accuracy. But the exploitation of two different modality sources remains a challenging issue. In this paper, inspired by recent works on attention networks that could focus on important regions and ignore unnecessary information, we propose a novel network: Channel-Spatial Attention Network (CSA6D) to estimate the 6D object pose from RGB-D camera. The proposed CSA6D includes a pre-trained 2D network to segment the interested objects from RGB image. Then it uses two separate networks to extract appearance and geometrical features from RGB and depth images for each segmented object. Two feature vectors for each pixel are stacked together as a fusion vector which is refined by an attention module to generate a aggregated feature vector. The attention module includes a channel attention block and a spatial attention block which can effectively leverage the concatenated embeddings into accurate 6D pose prediction on known objects. We evaluate proposed network on two benchmark datasets **YCB-Video** dataset and **LineMod** dataset and the results show it can outperform previous state-of-the-art methods under **ADD** and **ADD-S** metrics. Also, the attention map demonstrates our proposed network searches for the unique geometry information as the most likely features for pose estimation. From experiments, we conclude that the proposed network can accurately estimate the object pose by effectively leveraging multi-modality features.

Keywords 6D object pose estimation · Convolutional neural network · Feature fusion · Attention mechanism

Introduction

The aim to solve 6D object pose estimation problem with RGB or RGB-D images is to detect objects and estimate their orientations and translations relative to the given canonical models. It is a long standing problem in computer vision and robotics communities. Potentially the solutions to the problem could be applied to robot manipulation [1–3], self-driving cars [4, 5] or augmented reality [6, 7]. There are still some challenging issues in solving the problem when the images include severe occlusions, cluttered background, lighting variations, texture-less objects, or symmetrical objects.

Traditionally geometrical methods were used to solve the problem by matching RGB image features with object's 3D models [8, 9]. These methods require well-designed handcrafted features which are not robust to lighting variations, background clutters, or texture-less objects.

Recently deep learning methods have been proposed to solve the problem as CNNs have shown significant robustness to environment variations. Some of them took a holistic method to train end-to-end neural networks and regress the 6D pose directly from the networks [10]. Some of them exploited a key-point method and solved the problem with two stages: first estimate 2D keypoints of the object using deep networks and then estimate the 6D pose via 2D-3D correspondences with a PnP algorithm [11, 12]. Dense methods were also explored where a feature is extracted for each object pixel or patch and then the best minimal set of points

✉ Tao Chen
tc17339@essex.ac.uk

Dongbing Gu
dgu@essex.ac.uk

[1] School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, UK

Published online: 29 November 2021

is selected via the RANSAC algorithm or each feature casts a vote for 6D pose hypotheses [13].

RGB-D cameras have made two data modalities (RGB images and depth images) easily available and further pushed the research front for better 6D pose estimation. Some of RGB-D methods first estimated an initial pose from RGB image and then refined it on point clouds using an ICP algorithm or other optimization algorithms [14, 15]. Others used two separate networks for RGB images and 3D point cloud to extract appearance and point-wise geometrical features, then concatenated both features to regress the 6D pose [16].

Recently attention mechanisms have shown a remarkable performance in deep learning applications. The Transformer [17] becomes very effective in natural language processing tasks. In 2D image and 3D data tasks, it also demonstrates powerful capability to enhance the feature representation [18–21]. Woo et al [22] proposed an attention module (CBAM) that can process a given feature map in terms of spatial and channel dimensions to focus on necessary information. Both the transformer and the CBAM use attention mechanisms to enhance the relevance features while weaken the non-relevant features via generating weighted parameters. But the transformer generates weighted parameters based on the key-value-query concepts in information retrieval tasks while the CBAM generates them based on channel and spatial dimensions. For graph-structure data, Veličković et al [23] introduced a network architecture with an attention layer which considers the contribution from different neighbors of a node.

In this paper, we propose a novel end-to-end network: Channel-Spatial Attention Network (CSA6D) for 6D object pose estimation from RGB-D images. The proposed CSA6D includes a pre-trained 2D network to segment the interested objects from RGB image. Then it uses a 2D image detector and a 3D point cloud detector to extract appearance features and point-wise geometrical features from each segmented object. Two feature vectors for each pixel are stacked together as a fusion vector. Next it uses an attention module to process the fusion vector along spatial and channel axes to obtain an aggregated feature vector. Finally the 6D object pose is directly estimated from the aggregated feature vector via fully connected layers.

Our innovation is the use of an attention module to refine the fusion feature vector alone spatial and channel axes to improve the representation of feature map, and this design leads to a considerable accuracy improvement, so post-processing steps are unnecessary in our model. In previous work [16], the fusion feature vector is directly fed to stacked MLP layers to regress the output. Here we argue that this process might not exploit the potential of all the information well while our proposed attention module could focus on more important features for pose regression. Since two modality features are simply blending together, the spatial-attention and channel-attention blocks are used to extract related representative features from their embedding space while keeping the original structure. Specifically, this design makes a robust representation for the modality fusion scheme and does not require a costly refinement step.

We evaluate our model on the LindMod dataset [8] and YCB-Video dataset [10]. The quantitative result shows that our proposed model can achieve a result with the state-of-the-art accuracy compared with other learning methods.

Related Works

Classical methods for estimating the 6D object pose mainly rely on template matching techniques [8], which are sensitive to cluttered environments and lighting variations. Deep learning-based methods have demonstrated resilient capability to some challenging scenes. Here we review deep learning-based methods for 6D pose estimation.

RGB methods: Normally, the 2D bounding box and segment mask of an object are usually cropped from images, then taken as the input for CNN-based 6D pose estimation approaches. There are many deep learning architectures that achieved excellent performance in object detection [24, 25] and segmentation tasks [26]. SSD6D [27] utilized a SSD detector to find the interested objects first, then their viewpoints were roughly approximated through classification instead of regressing 6D pose directly. BB8 [28] took a holistic approach where the 6D pose is directly regressed by a network following from a segmentation network. Keypoint-based methods [29, 30] first estimated some sparse 2D keypoints of an object and then estimated the 6D pose through 2D-3D correspondences with a PnP algorithm [31]. However, keypoint-based methods are not very effective when some keypoints are occluded. DPOD estimated dense multi-class 2D-3D correspondence maps and allowed for more robust estimation [1]. Pix2Pose [32] used an auto-encoder to make pixel-wise prediction for object's 3D coordinate, obtaining the pose through solving 2D-3D correspondence using the PnP algorithm. They also attempted to use a Generative Adversarial Network [33] to recover the occlusion part of the object. To consider each pixel's contribution, PVNet [34] made use of a voting scheme to select the best keypoint from each pixel's prediction. PoseCNN [10] proposed a network that regresses the center of object and regresses the 3D center distance from the camera directly. TrackNet [35] and DeepIM [36] captured the discrepancy between the current and previous images and used a network to estimate the pose residuals. But they required an initial pose to be estimated at the start of the process in order to make iterative refinement. These

RGB methods are lack of geometrical (depth) information, which limits the performance of 6D pose estimation.

RGB-D methods: RGB-D data provides additional geometrical information along with appearance information, which offers an opportunity to explore 3D point clouds for 6D pose estimation. The work in [14] estimated an initial 6D pose using RGB method and refined the pose iteratively via point cloud with an ICP algorithm [37]. PointNet [38] pioneered the work in point-wise feature extraction for classification and segmentation on 3D data. DenseFusion [16] proposed to use two separate networks to extract features from RGB image and depth image, and a CNN to regress the 6D pose followed by an ICP-like algorithm. In their work, the pose is estimated from the concatenated features of RGB image and depth image. Gao et al [39] directly utilized two PointNet-like networks to regress the 6D pose from unordered point sets. FFB6D extended Densefusion with a full flow bidirectional fusion network and used appearance information in RGB image and geometry information in depth image as complementary information during their feature extraction [40].

Attention mechanisms: Visual attention mechanisms are able to focus on certain parts of an image while perceiving the surrounding region through a correlation vector. It could enhance the global view by using the correlation weights and improve the model accuracy. Recently they are successfully applied for visual tasks, such as image classification [18, 19], image captioning [21], scene segmentation [20]. Spatial and channel attentions provide a mechanism to focus where to look in a spatial attention block and what to look in a channel block [22, 41]. Using spatial attention only for 6D pose estimation was proposed in [42]. It works under an iterative refinement framework like DeepIM[36] where the pose residuals are estimated from a network. A graph attention module was added after the feature extraction in [43] for the 6D pose estimation from RGB-D images.

In this paper, we will fuse appearance features in RGB image and geometry features in depth image together, and use both spatial and channel attention blocks to refine the fusion feature vector along spatial and channel axes to improve the representation of the feature map for 6D pose estimation.

Methods

In this section, we will describe our network CSA6D in details. Our final goal is to estimate the rotation matrix $R \in \mathbb{R}^{4\times1}$ in quaternion and translation vector $T \in \mathbb{R}^{3\times1}$ of detected objects. We use RGB-D images as input and no refinement or post-processing step required in our system.

Model Architecture

The architecture of our CSA6D is depicted in Fig. 1. An input RGB-D image with 640 x 480 pixels is fed to the system. Firstly, the RGB image is segmented by using a semantic segmentation network and each interested object is cropped from the image with its corresponding 2D bounding box and masks. By finding the corresponding region in the depth image with object masks, the object 3D point cloud is recovered by the camera calibration matrix and cropped depth region. The pre-trained segmentation network we used is an encoder-decoder network Mask R-CNN [26]. This segmentation network outputs $N + 1$ binary maps in which each pixel belonging to that class (background class included) is activated and N is the number of object classes. The image

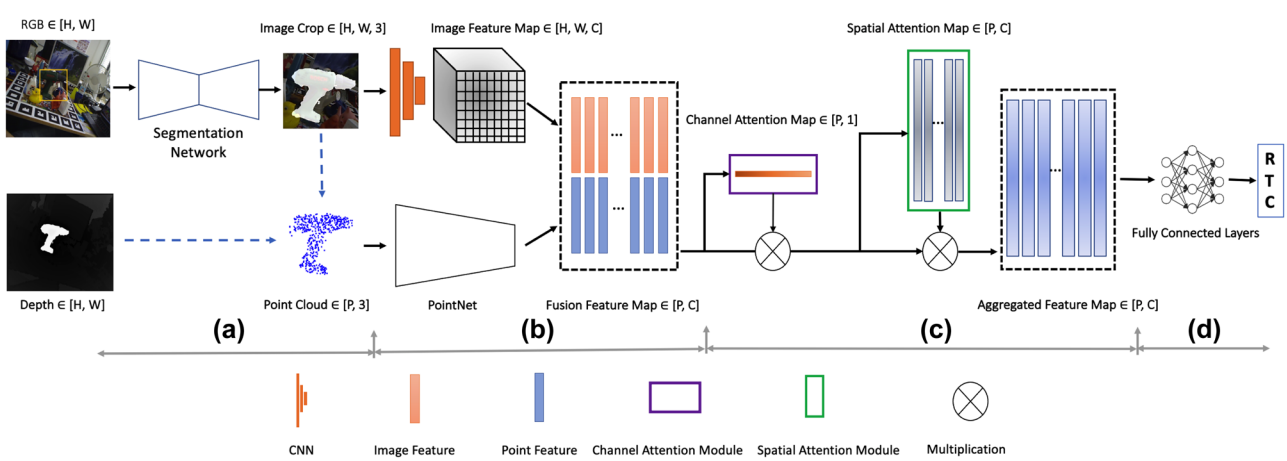

Fig. 1 Overview of our proposed framework: (**a**) image segmentation and point cloud re-construction, (**b**) appearance and geometrical feature extraction, and feature fusion, (**c**) attention module for feature refinement, and (**d**) pose regression. The inputs are RGB image and depth image. The final outputs are **R**:rotation, **T**:translation, **C**:confidence

patch that contains interested object is cropped by using 2D bounding box obtained from the segmentation network.

Secondly, we extract appearance features of the object from the cropped image patch using a CNN. Here we use Pyramid Scene Parsing Network (PSPNet) [44] as the appearance feature extractor to obtain an image feature map shown as the top branch in Fig. 1. The resulted feature map has size $H \times W \times C$ where C represents the dimension of each pixel in their feature space. H and W are the height and width of the original image patch. We extract geometry features from the 3D point cloud data using a variant of PointNet [38] shown as the bottom branch in Fig. 1. The correspondence between two features for each pixel is established by using projection.

Thirdly, as appearance features in RGB image and geometry features in depth are complementary, they are stacked together as a fusion vector to form a compact representation of the interested object. We apply an channel attention block followed by a spatial attention block to refine the fusion feature. More specifically, the attention blocks perform max-pool and average-pool operations in channel and spatial axes to get a new aggregated feature vector that has same dimensionality with the fusion feature vector.

Finally we have three separate branches to estimate the rotation, translation and confidence, respectively, each of them using five fully connected layers. The confidence score refers to the confidence the network has on each prediction.

Attention Module

Due to the occlusion of objects or potential segmentation errors, we might include the pixels that belong to other objects or background. This result could deteriorate the robustness of fusion features. To overcome this problem, our attention module is to refine the fusion features so that it could alleviate the potential problem. Our attention module comprises of two blocks, channel attention block and spatial attention block and this is inspired by CBAM [22]. They are modified to process 1D fusion features used in our network instead of 2D image features originally proposed.

Assuming a fusion feature F has a shape $\mathbb{R}^{P \times C}$ where P is the number of pixels, the channel attention block can produce an 1D channel attention vector $M_c \in \mathbb{R}^{P \times 1}$, the spatial attention block can refine a new spatial attention feature $M_s \in \mathbb{R}^{P \times C}$. These two blocks are concatenated together as shown in Fig. 1. The channel attention block takes the fusion feature as input and generate a channel attention feature F'. These two features are multiplied together and the result is fed to the spatial attention block to generate a spatial attention feature. Again these two features are multiplied together to generate an aggregated feature vector F''. Hence, the overall procedure can be written as follows:

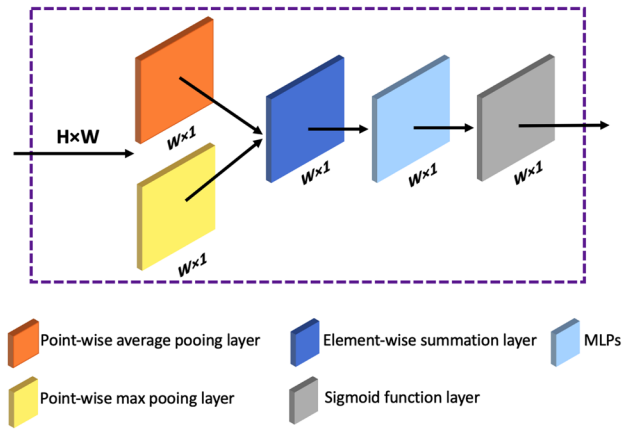

Fig. 2 The channel-attention block. $H \times W$ represents the input dimensions. Operations are stated inside the box and the feature dimensions are shown after the operation box. Multi-layer perceptron(MLP) has three layers with output dimensions (W, W/16, W)

$$\begin{aligned} F' &= M_c(F) \bigotimes F \\ F'' &= M_s(F') \bigotimes F' \end{aligned} \tag{1}$$

where F' is the output from the channel attention block and F'' is the final output that has the same shape with the fusion feature. $\bigotimes$ represents the element-wise multiplication. Broadcast operation to attention map is applied if needed.

Channel Attention Block

The details of the channel attention block are shown in Fig. 2. It applies point-wise max-pool and average-pool operations to the fusion feature map, respectively, and the resulted descriptors are summed element-wise. A shared multi-layer perceptron network is used to process the resulted descriptor, which has three neuron layers. To prevent the network's parameters overload, the middle neuron size is set to $W/16$ that is suggested by [22]. The feature map generated by the MLP has dimension $W \times 1$ and is processed by the Sigmoid function to produce the final channel attention map with size $W \times 1$.

Spatial Attention Module

After the first multiplication shown in Fig. 1 (to enable the multiplication, broadcasting operation is applied to the channel attention map), the channel attention vector is refined as $F' \in \mathbb{R}^{H \times W}$. To get the spatial attention feature, max-pool and average-pool operations are applied to generate two features F'_{avg} and F'_{max}, and both have size $W \times 1$, and they are concatenated. Average-pooling and Max-pooling are commonly used pooling functions. The intuition behind

these two pooling functions is that combining the global information captured by average-pool function and the local information captured by max-pool function can have a better performance for our task than using one of them. Here, we use a 1×1 convolutional layer to process the concatenated feature instead of the convolutional layer with kernel size of 7×7, then followed by batch-normalization and ReLu operations. So the spatial attention block is calculated as:

$$\begin{aligned} F'_{avg} &= AvgPool(F') \\ F'_{max} &= MaxPool(F') \\ M_s(F') &= sigmoid(f^{1\times1}(F'_{avg}, F'_{max})) \end{aligned} \tag{2}$$

where *sigmoid* denotes the Sigmoid function, which is used to output the normalized feature. Finally the aggregated feature vector F'' is obtained by multiplication, and used to estimate the object pose by the pose predictor.

Loss Function

In this subsection, we describe the loss function used in our model. we train the loss in a mean square error function as shown below:

$$L_i = \frac{1}{m} \sum_j \left\| (\mathbf{T}x_j - \tilde{\mathbf{T}}x_j) \right\|^2 \tag{3}$$

where x_j is the jth point randomly selected from points of object model, and $\mathbf{T}$ is the ground truth transformation and $\tilde{\mathbf{T}}$ is the predicted transformation from j^{th} refined attention features. We also output the confidences of model's predictions, which we would like to utilize to penalize the bad features. So inspired by the DenseFusion [16], we add a regularize term to balance overall prediction. Hence, our final loss function is described as:

$$L = \frac{1}{N} \sum_i (L_i C_i - Wlog(C_i)) \tag{4}$$

where N is the total number of sampled refined attention features, and W is the hyperparameter for confidence. During inference, the highest confidence is selected as final output.

Experiments

In this section, we describe the training details of our network. The network is evaluated on challenging datasets **YCB-Video** dataset [10] and **LindMod** dataset [8] . We use a GeForce GTX 1080 Ti graphic card to train our network, which took appx. 300 hours to finish the iterations of 500 epochs on the YCB-Video dataset, and on the LineMod dataset it costs appx. 200 hours to finish 500 epochs. The network is implemented in Pytorch.

Datasets

The LineMod and YCB-Video datasets are two commonly used benchmark datasets. The YCB-Video dataset contains mixed 21 textured and texture-less household objects coming from 92 video sequences. Each frame is annotated with 6D object pose ground-truth. The LineMod dataset has 13 texture-less objects placed on the table in the cluttered background. The datasets were captured by Kinect camera, and each image has its associated depth image and has an object pose annotation. The spilt of training/test sets are unchanged with official datasets.

Training

In the semantic segmentation network for appearance feature learning, *ResNet-18* [46] is used as backbone network, and 4 pyramid levels for pooling are 1×1, 2×2, 3×3, and 6×6. The dimension of geometry feature is set to 1024 and the dimension of appearance feature is 384, hence, the dimension of fusion feature is 1024 + 384. To predict the pose, we have three independent Multi-layer perceptrons (MLP) applied on the aggregated feature in which each MLP has 5 hidden neuron layers, (1408-640-256-128-4) are the size of hidden layer for the rotation prediction (quaternion), (1408-640-256-128-3) for the translation prediction and (1408-640-256-128-1) for the confidence prediction. In order to prevent over-fitting, we apply data argumentation technique on input RGB patch. For instance, we add some random noises to brightness, contrast, saturation and hue of image of training set. In point cloud, tiny translation error is added. To balance the accuracy and computation, we use Farthest Point Sampling (FPS) algorithm proposed in PointNet to sample 1000 points from the recovered point cloud before feeding it to PointNet. In this way, we can maintain the surface information with limited number of points. The hyperparameter W in equation 4 is chosen as 0.01.

Evaluation Metrics

ADD(S) Metrics

To evaluate the network's performance, we use Average Distance of Model Points (ADD) [10] as metric to non-symmetric objects and Average Closest Point Distance (ADD-S) to symmetric objects.

$$ADD = \frac{1}{m} \sum_{x \in M} \left\| (\mathbf{R}x + \mathbf{T}) - (\tilde{\mathbf{R}}x + \tilde{\mathbf{T}}) \right\|_2$$

where $\tilde{\mathbf{R}}$ and $\tilde{\mathbf{T}}$ are the predicted rotation and translation matrices, while $\mathbf{R}$ and T are the ground-truth of matrices. x denotes the points randomly selected from 3D model of object of interest. The prediction of rotation and translation

is considered as correct if the score of ADD is lower than a predefined threshold. To evaluate the model's performance on symmetric objects, the ADD-S is used for evaluation. The ADD-S score is calculated as average distance to the closest point.

$$\text{ADD-S} = \frac{1}{m} \sum_{x_1 \in M} \min_{x_2 \in M} \left\| (\mathbf{R}x_1 + \mathbf{T}) - (\tilde{\mathbf{R}}x_2 + \tilde{\mathbf{T}}) \right\|_2$$

where x_1 and x_2 are selected from the same 3D model.

In this paper, we report the area under curve (AUC) for ADD and ADD-S metrics. Also, we set the maximum threshold of both curves to be 0.1m. Beside this, we further test the ADD-S under threshold 0.01m to illustrate the network's tolerance to small errors. During evaluation, we use ADD metric for non-symmetric objects and ADD-S for symmetric objects.

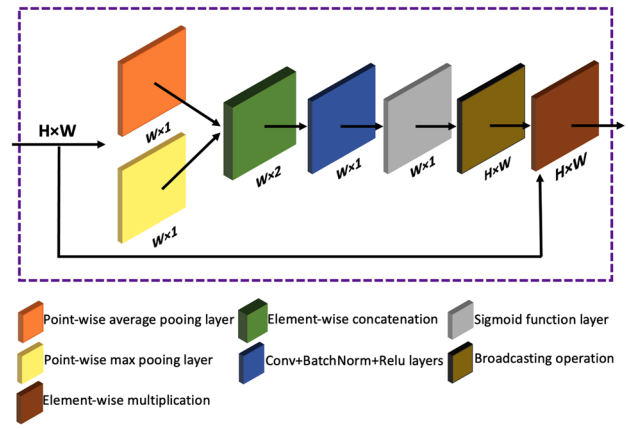

Fig. 3 The spatial-attention block. The operations inside the box are 1D convolution, batch-normalization, and ReLu function. Broadcasting operation duplicates its input feature map $W \times 1$ for H times to form a feature map with dimension $W \times H$

2D Re-Projection Error

In addition to ADD(S) metrics, we also use the 2D projection metric to quantify the performance of our network. In this way, the object model points are projected to image plane by ground-truth pose and predicted pose. The prediction pose is treated as correct if the average distance of corresponding points is less than 5 pixels. The 2D Re-projection error can be calculated as below:

Table 1 Quantitative evaluation result on the YCB-Video dataset. Objects with bold indicate symmetry objects. Numbers with bold mean the best performance in comparison

	RGB		**RGB-D**							
	PoseCNN [10]		**PoseCNN** [10]**+ICP**		**PointFusion** [45]		**DenseFusion** [16]		**Ours**	
Objects	**ADD-S**	**ADD(S)**	**ADD-S**	**ADD(S)**	**ADD-S**	**ADD(S)**	**ADD-S**	**ADD(S)**	**ADD-S**	**ADD(S)**
002_master_chef_can	84.0	50.9	95.8	69.0	90.9	-	**96.4**	73.1	95.8	71.8
003_cracker_box_vi	76.9	51.7	91.8	80.7	80.5	-	95.5	94.0	**96.4**	95.3
004_sugar_box	84.3	68.6	**98.2**	97.2	90.4	-	97.5	96.5	97.6	97.0
005_tomato_soup_can	80.9	66.0	94.5	81.6	91.9	-	**94.6**	85.4	94.5	85.9
006_mustard_bottle	90.2	79.9	**98.4**	97.0	88.5	-	97.2	94.7	97.9	96.5
007_tuna_fish_can	89.7	70.4	**97.1**	83.1	93.8	-	96.6	81.7	96.8	81.4
008_pudding_box	79.0	62.9	**97.9**	96.6	87.5	-	96.5	93.2	96.4	93.7
009_gelatin_box	87.1	75.2	**98.8**	98.2	95.0	-	98.1	96.6	98.3	97.5
010_potted_meat_can	78.5	59.6	**92.8**	83.8	86.4	-	91.3	83.5	91.7	82.5
011_banana	85.9	72.3	**96.9**	91.6	84.7	-	96.6	83.1	96.4	87.0
019_pitcher_base	76.8	52.5	**97.8**	96.7	85.5	-	97.1	96.8	**97.8**	97.0
021_bleach_cleanser	71.9	50.5	**96.8**	92.3	81.0	-	95.8	90.2	96.0	91.1
024_bowl	69.7	69.7	78.3	78.3	75.7	75.7	88.2	88.2	**88.6**	**88.6**
025_mug	78.0	57.7	95.1	81.4	94.2	-	**97.1**	88.9	96.7	93.0
035_power_drill	72.8	55.1	**98.0**	96.9	71.5	-	96.0	92.8	96.4	95.0
036_wood_block	65.8	65.8	90.5	90.5	68.1	68.1	89.7	89.7	**93.2**	**93.2**
037_scissors	56.2	35.8	92.2	78.4	76.7	-	**95.2**	77.8	90.2	90.2
040_large_marker	71.4	58.0	97.2	85.4	87.9	-	97.5	92.9	**97.6**	92.2
051_large_clamp	49.9	49.9	**75.4**	**75.4**	65.9	65.9	72.9	72.9	73.7	73.7
052_extra_large_clamp	47.0	47.0	65.3	65.3	60.4	60.4	69.8	69.8	**70.8**	**70.8**
061_foam_brick	87.8	87.8	**97.1**	**97.1**	91.8	91.8	92.5	92.5	95.7	95.7
All	75.9	53.7	93.0	79.3	83.9	-	93.1	87.3	**93.3**	**89.0**

Table 2 Quantitative evaluation result on the YCB-Video dataset. Objects with bold indicate symmetry objects. Numbers with bold mean the best performance in comparison

	RGB-D					
	DenseFusion [16]	**PoseCNN** [10]**+ICP**	**SSD-6D** [27]**+ICP**	**PVNet** [34]	**Ours**	**Ours(ADD(S)< 0.01m)**
ape	92.3	77.0	65	43.6	**94.6**	94.6
bench vi	93.2	97.5	80	**99.9**	96.9	92.2
camera	94.4	93.5	78	86.8	**98.7**	96.4
can	93.1	96.5	86	95.4	**97.3**	94.5
cat	96.5	82.1	70	79.3	**99.3**	97.9
driller	87.0	95.0	73	96.4	**97.1**	86.5
duck	92.3	77.7	66	52.5	**97.0**	95.3
egg box	99.8	97.1	**100**	99.1	99.8	97.1
glue	**100.0**	99.4	**100**	95.6	99.7	97.3
hole punch	92.1	52.8	49	81.9	**97.3**	94.1
iron	97.0	98.3	78	**98.8**	98.6	90.8
lamp	95.3	97.5	73	**99.3**	98.5	95.4
phone	92.8	87.7	79	92.4	**97.7**	93.5
mean	94.3	88.6	79	86.2	**97.9**	94.3

$$2D\_Reproj = \frac{1}{m}\sum_{x\in M}\left\|K(\mathbf{R}x+\mathbf{T})-K(\tilde{\mathbf{R}}x+\tilde{\mathbf{T}})\right\|_2$$

where K is the camera intrinsic matrix.

Results

YCB-Video Dataset

In this section, we first report the evaluated result of our network on the YCB-Video dataset. We also compare our network with four state-of-the-art pose estimation algorithms (PoseCNN [10], PoseCNN [10] with ICP refinement, PointFusion [45], and DenseFusion [16]). As we can see in Table 1, the algorithms are classified into **RGB** class and **RGB-D** class. Clearly, the **RGB** method PoseCNN is lack of accuracy compared with other methods, no matter under which evaluation metrics. We believe this is due to the loss of geometry information. By using the result of PoseCNN as initial estimation, the refinement algorithm ICP can largely improve the performance through optimizing the initial estimation in 3D space. PointFusion [45] and DenseFusion [16] both used RGB image and depth image as their inputs and they can extract appearance and geometrical features for pose estimation. Compared with these two RGB-D methods, our model completely outperform the PointFusion in terms of the performance of individual object or average performance under the ADD-S and ADD(S). Evaluating by ADD-S metric, we lead DenseFusion 0.2% in the performance for all objects, 1.7% under ADD(S). Also, we have more number of highest score objects compared with DenseFusion. It is worth noting that our method has 3 out of 5 best performances on symmetry objects (with bold name in Table 1). As we know, symmetry objects could cause ambiguity for the feature learning. Hence, we can conclude that our result shows a strong capability of the proposed attention module in learning effective representation from those symmetry objects.

Table 3 Quantitative Evaluation result On the LineMod dataset with 2D projection metrics

	ape	bench vi	camera	can	cat	riller	duck	**egg box**	**glue**	hole punch	iron	lamp	phone	mean
DenseFuion(10 pixels)	96.8	92.5	97.7	94.0	97.1	86.5	95.9	**100**	**99.9**	94.9	97.0	95.5	94.6	95.6
Ours(10 pixels)	**98.6**	**95.8**	**98.8**	**97.4**	**99.5**	**95.1**	**98.4**	99.9	**99.9**	**98.2**	**97.8**	**97.5**	**97.6**	**98.1**
DenseFuion(5 pixels)	92.9	85.0	85.0	90.8	95.3	76.2	90.7	**99.7**	99.6	86.9	87.4	91.0	87.6	89.9
Ours(5 pixels)	**94.7**	**90.2**	**94.5**	**94.1**	**98.2**	**85.4**	**94.5**	99.5	**99.9**	**94.0**	**89.8**	**93.3**	**92.2**	**93.9**
DenseFuion(2 pixels)	60.2	48.0	35.3	64.4	66.3	37.0	55.1	67.5	89.1	45.6	**41.0**	56.4	46.6	55.0
Ours(2 pixels)	**63.3**	**51.3**	**49.7**	**67.0**	**77.7**	**37.8**	**61.4**	**68.4**	**91.9**	**56.4**	40.2	**62.2**	**48.4**	**60.0**

LineMod Dataset

We report the evaluated result of our network on the LineMod dataset. We also compare our network with four state-of-the-art pose estimation algorithms (DenseFusion [16], PoseCNN [10], SSD-6D [27], and PVNet [34]). To achieve a fair comparison, all segmented masks used in these methods are provided by PoseCNN. As we can see in Table 2, our method outperforms other methods. Ours refer to the evaluation result using AUC threshold under 0.1m. Our method leads DenseFusion algorithm 3.6% and outperforms PoseCNN nearly 9%. Even we use more strict criteria (ADD-S<0.01m), our method achieves an equivalent performance with DenseFusion 94.3% and still outperforms PoseCNN. For the individual object, while DenseFusion has 100% accuracy on *glue*, we achieve the highest prediction on 8 out of 13 objects.

In Table 3, the accuracy results by evaluating of 2D projection metric are shown. As DenseFusion does not provide its evaluation result, so we re-trained it to obtain the statistical result shown in Table 3. We evaluate the model in three different thresholds(10 pixels, 5 pixels and 2 pixels). Under the condition of 10 pixels, our network has the highest accuracy almost for every objects, except the egg box object, but the gap between them is small(0.1%) and glue object with the same accuracy. In 5 pixel criteria, We see our network has the highest accuracy almost for every objects, except the egg box object, but the gap between them is 0.2%. When the threshold decreases to 2 pixels, both methods' accuracy drop sharply. But our network still has the relative better performance.

We also test our network's performance within small average distance thresholds (ranged 0 - 0.01meter). In this way, we can see how well our model is in the high-precision pose estimation tasks. In Fig. 4, we report the accuracy of each object in LineMod with varying threshold. As we can see that until the threshold of 0.006 meters, our network can achieve an accurate prediction (>80%). Less than threshold of 0.005, the accuracy curves drop sharply. Note that the object egg box has poor prediction when the threshold is low than 0.07. This situation may be caused by the hard prediction of symmetric object in small tolerance of error of ADD-S metric. In Fig. 5, we report the average accuracy of all objects in the LineMod dataset for DenseFusion and ours. For threshold > 0.03, our curve is in the above of curve of DenseFusion, which means our network has a better performance. Some samples of our estimation results are shown in Fig. 6 by projecting their estimated poses back to the image. They provide a clear view on the good quality of our estimation results.

Specifically, we draw the attention maps as shown in Fig. 8, where specified region is highlighted as important area for object's pose. The darker the color, the more crucial the area. For instance, in the top row object kettle (object can in Tables 2 and 3) is highlighted in its handle area and this region has the highest confidence to object pose. In the second row object driller and fourth row object lamp, their heads are being treated as the parts that have the best estimation for their pose, and we believe this is due to their heads' distinct geometrical information. Furthermore, our model identifies the edge of symmetry object egg box as its focused region which is much reasonable.

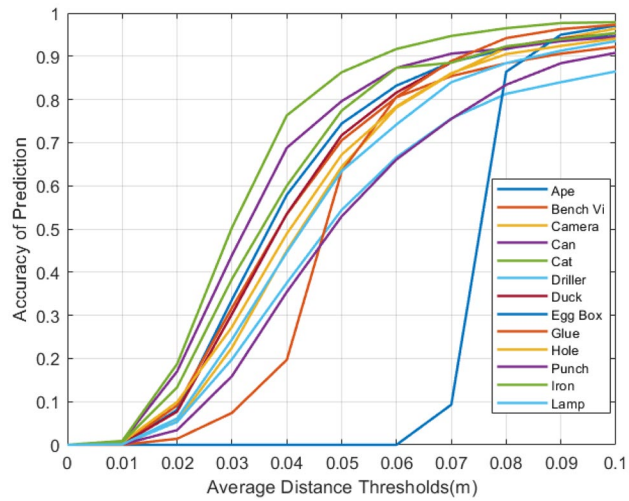

Fig. 4 Accuracy–threshold curve for each object in LineMod

Ablation Study

To investigate how our attention modules affect the performance, we test our model with different setups in the LineMod dataset. As shown in Table 4, Channel Block and

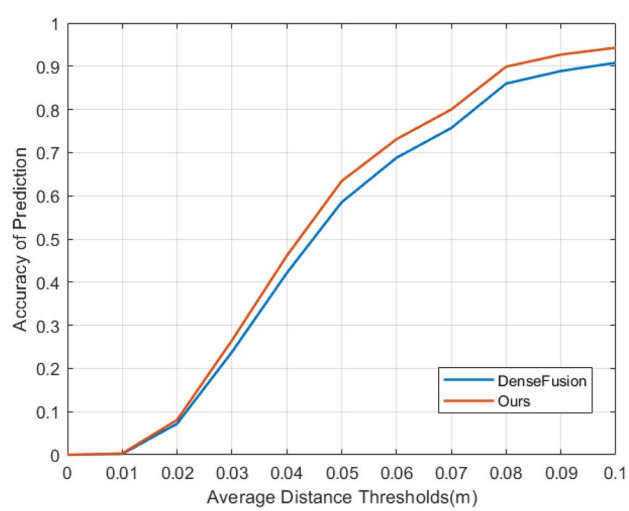

Fig. 5 Average accuracy by varying average distance thresholds

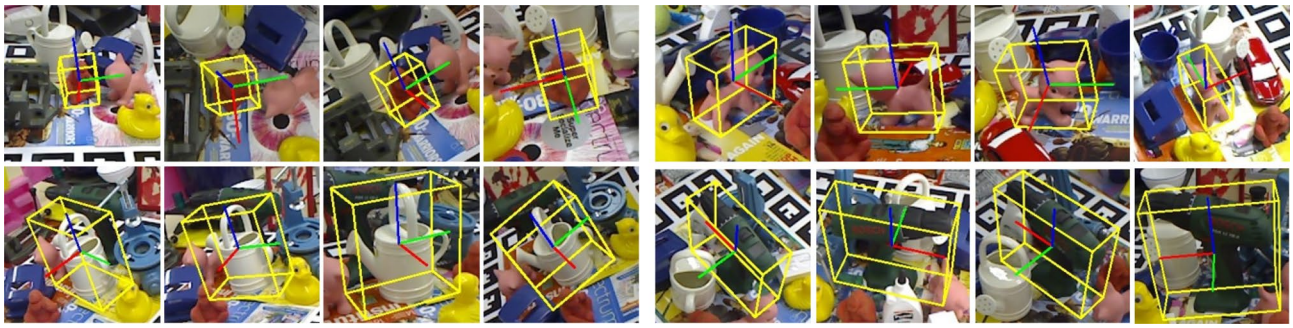

Fig. 6 Quality results on the LineMod dataset

Table 4 Ablation study

	Time(ms/image)	Parameters	System parameters(Millions)	ADD(S)	ADD-S
Channel Block	18.1	2184	23.624M	96.5	99.5
Spatial Block	**17.7**	**16**	**23.374M**	97.2	99.6
Channel+Spatial Blocks	18.3	2200	23.624M	**97.9**	**99.7**

Spatial Block indicate that only corresponding attention block is used in our model, and Channel + Spatial Blocks refers to our complete framework. From the perspective of estimation accuracy, the configuration of Channel + Spatial Blocks show the best accuracy in terms of ADD(S) and ADD-S metrics. But it also has the longest inference time (18.2 milliseconds per image) for each input image and the largest number of parameters in memory. The **parameters** column indicates the parameters for block itself. In the contrary, the **System parameters** column means the full number of parameters of our model with the existence of corresponding block. As we can see from the table, the Spatial block only has 16 parameters, which is quite tiny compared with the Channel block, and it makes sense that the Spatial block can run with the fastest inference time. Note that the number of parameters in the Spatial block contributes much less to the entire model, because we only have learnable parameters in two convolutional layers as depicted in Fig. 3. The Channel block has almost the same system parameters with the Channel + Spatial blocks. In summary, the combination of Channel block and Spatial block do improve the accuracy and they are lightweight compared with the entire model.

We believe that the potential of this attention module could also be used in object pose tracking tasks with a framework of pose refinement that predicts the residual of pose within two consecutive frames. As indicated in Fig. 8, our model could focus on some particular regions of the object for pose estimation. This might improve the tracking performance when the occlusion occurs in some scenes.

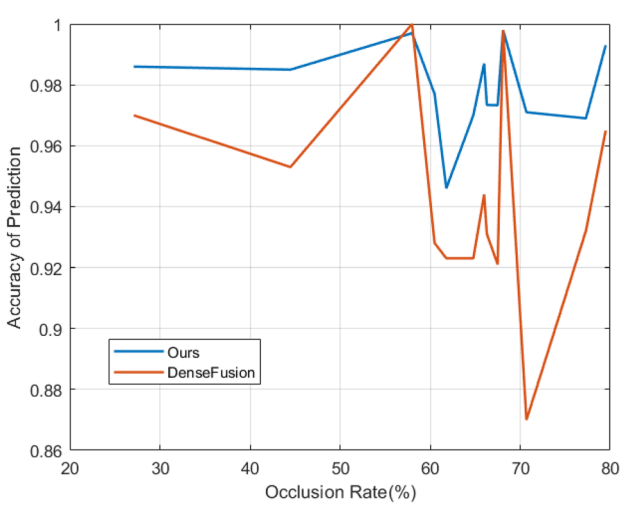

Fig. 7 Prediction accuracy against object occlusion rate

Robustness to Occlusion

To explore how robust of our network with occlusion objects, we proposed a *occlusion rate* to reflect how much of an object being occluded. We take p as the total number of pixels of an interested object in ground truth data, and λ as the number of pixels being projected by object model with ground truth pose. Due to self-occlusion of the object, we treat the number of pixels projected in image as $\lambda/2$. So the occlusion rate r can be represented as below:

$$r = 1 - p/(\lambda/2)$$

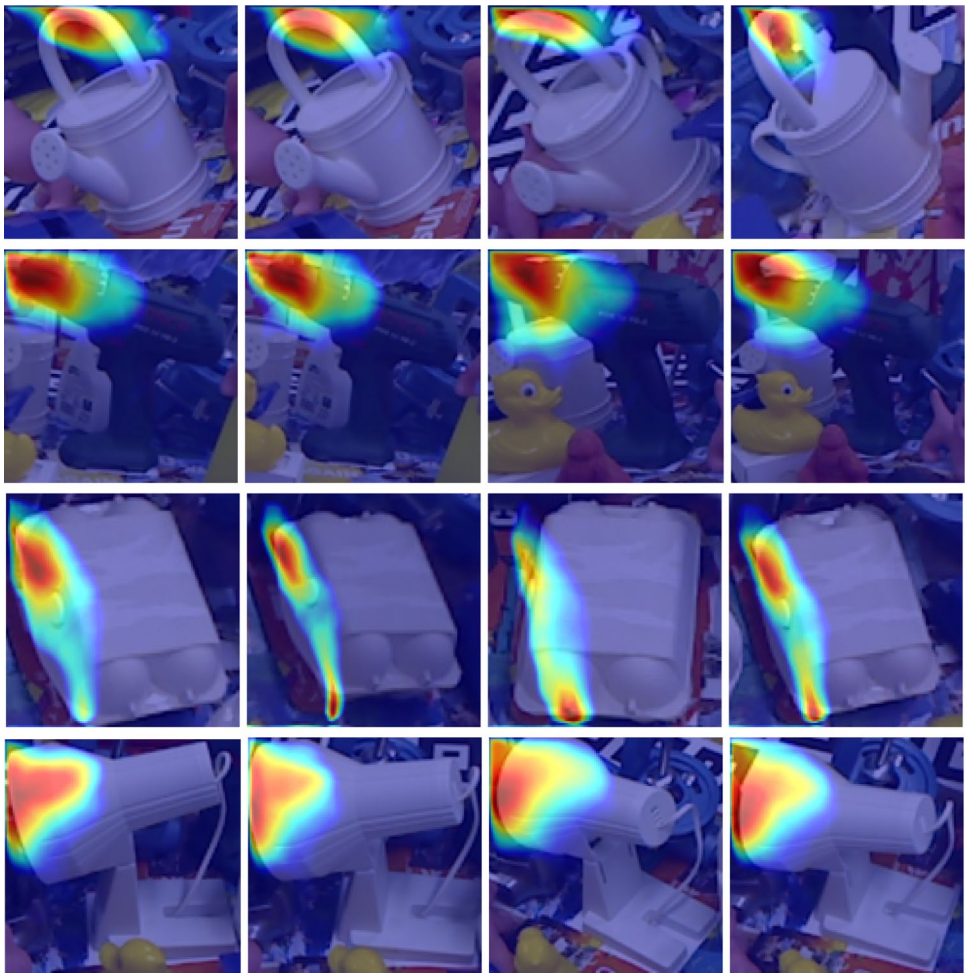

Fig. 8 Visualization of attention regions

Therefore, the bigger values of r, the more occlusion of an object. In experiment, we calculated r for each labeled object in the LineMod dataset and averaged them. In Fig. 7, we show the performance of our network against different occlusion rates. The blue curve represents our network's estimation accuracy in terms of different occlusion rates. When the r increases, our accuracy remains stable but the curve of DenseFusion (Orange color) has some fluctuations as r increases.

Conclusions

In this work, we present a network CSA6D that can estimate the 6D object pose from RGB-D image. Both appearance features from RGB image and geometry features from depth image are densely fused together for direct pose regression. Our main innovation includes the use of channel and spatial attention modules to refine the dense fusion feature in order to improve the network performance without adding too much computational burden. Our evaluation results on public datasets show that our network is accurate and robust compared with some existing methods.

The attention module is lightweight and efficient and could be easily inserted into other leaning tasks. We demonstrate that our model can extract features of specific regions for object pose estimation tasks. In our future work, we aim to reduce the computational complexity further in real-time applications. Based on our ablation study of attention blocks, the inference time of our model can be reduced using the spatial attention block only without sacrificing too much accuracy. Also, we believe that making our model (especially the image feature extraction model) lighter could significantly reduce the inference time, which can make it possible to work in real-time applications.

Appendix

6D	6 Degrees of Freedom
ADD	Average Distance
ADD-S	Average Distance to the Closest Point.
AUC	Area Under Curve
CSA6D	Channel-Spatial Attention Network
CNN	Convolutional Neural Network
FPS	Farthest Point Sampling
ICP	Iterative Closest Point
MLP	Multi-Layer Perceptron
PnP	Perspective-n-Point
RANSAC	Random Sample Consensus

Funding Information No funding was received in this work.

Declarations

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

Conflict of Interest The authors declare that they have no conflict of interest.

References

1. Tremblay J, To T, Sundaralingam B, Xiang Y, Fox D, Birchfield S. Deep object pose estimation for semantic robotic grasping of household objects. arXiv preprint arXiv:180910790. 2018.
2. Collet A, Martinez M, Srinivasa SS. The MOPED framework: Object recognition and pose estimation for manipulation. Int J Robot Res. 2011;30(10):1284–306.
3. Mahler J, Matl M, Liu X, Li A, Gealy D, Goldberg K. Dex-Net 3.0: Computing robust vacuum suction grasp targets in point Clouds using a new analytic model and deep learning. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). 2018. p. 5620-7.
4. Chen X, Ma H, Wan J, Li B, Xia T. Multi-view 3D object detection network for autonomous driving. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition; 2017. p. 1907-15.
5. Geiger A, Lenz P, Urtasun R. Are we ready for autonomous driving? the KITTI vision benchmark suite. In IEEE Conference on Computer Vision and Pattern Recognition. 2012;2012:3354–61.
6. Tan DJ, Navab N, Tombari F. 6D object pose estimation with depth images: a seamless approach for robotic interaction and augmented reality. arXiv preprint arXiv:170901459. 2017.
7. Marchand E, Uchiyama H, Spindler F. Pose Estimation for Augmented Reality: A Hands-On Survey. IEEE Trans Vis Comput Graph. 2015;22(12):2633–51.
8. Hinterstoisser S, Holzer S, Cagniart C, Ilic S, Konolige K, Navab N. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In IEEE International Conference on Computer Vision. 2011;2011:858–65.
9. Brachmann E, Krull A, Michel F, Gumhold S, Shotton J, Rother C. Learning 6D object pose estimation using 3D object coordinates. In: European Conference on Computer Vision. Springer; 2014. p. 536-51.
10. Xiang Y, Schmidt T, Narayanan V, Fox D. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. arXiv preprint arXiv:171100199. 2017.
11. Pavlakos G, Zhou X, Chan A, Derpanis KG, Daniilidis K. 6-DoF object pose from semantic keypoints. In IEEE International Conference on Robotics and Automation (ICRA). 2017;2017:2011–8.
12. Suwajanakorn S, Snavely N, Tompson J, Norouzi M. Discovery of latent 3D keypoints via end-to-end geometric reasoning. arXiv preprint arXiv:180703146. 2018.
13. Kehl W, Milletari F, Tombari F, Ilic S, Navab N. Deep Learning of local RGB-D patches for 3D object detection and 6D pose estimation. In: European Conference on Computer Vision. Springer; 2016. p. 205-20.
14. Jafari OH, Mustikovela SK, Pertsch K, Brachmann E, Rother C. iPose: Instance-aware 6D pose estimation of partly occluded objects. In: Asian Conference on Computer Vision. Springer; 2018. p. 477-92.
15. Labbé Y, Carpentier J, Aubry M, Sivic J. CosyPose: Consistent multi-view multi-object 6D pose estimation. In: European Conference on Computer Vision. Springer; 2020. p. 574-91.
16. Wang C, Xu D, Zhu Y, Martín-Martín R, Lu C, Fei-Fei L, et al. DenseFusion: 6D object pose estimation by iterative dense fusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2019. p. 3343-52.
17. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: Advances in Neural Information Processing Systems; 2017. p. 5998-6008.
18. Hu J, Shen L, Sun G. Squeeze-and-Excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2018. p. 7132-41.
19. Wang F, Jiang M, Qian C, Yang S, Li C, Zhang H, et al. Residual attention network for image classification. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 3156-64.
20. Fu J, Liu J, Tian H, Li Y, Bao Y, Fang Z, et al. Dual attention network for scene segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2019. p. 3146-54.
21. Xu K, Ba J, Kiros R, Cho K, Courville A, Salakhudinov R, et al. Show, attend and tell: neural image caption generation with visual attention. In: International Conference on Machine Learning. PMLR; 2015. p. 2048-57.
22. Woo S, Park J, Lee JY, Kweon IS. CBAM: convolutional block attention module. In: Proceedings of the European Conference on Computer Vision (ECCV); 2018. p. 3-19.

23. Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y. Graph Attention Networks. arXiv preprint arXiv:171010903. 2017.
24. Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2016. p. 779-88.
25. Ren S, He K, Girshick R, Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Adv Neural Inf Proces Syst. 2015;28:91–9.
26. He K, Gkioxari G, Dollár P, Girshick R. Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision; 2017. p. 2961-9.
27. Kehl W, Manhardt F, Tombari F, Ilic S, Navab N. SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again. In: Proceedings of the IEEE International Conference on Computer Vision; 2017. p. 1521-9.
28. Rad M, Lepetit V. BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth. In: Proceedings of the IEEE International Conference on Computer Vision; 2017. p. 3828-36.
29. Tekin B, Sinha SN, Fua P. Real-time seamless single shot 6D object pose prediction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2018. p. 292-301.
30. Hu Y, Hugonot J, Fua P, Salzmann M. Segmentation-driven 6D object pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2019. p. 3385-94.
31. Lepetit V, Moreno-Noguer F, Fua P. EPnP: an accurate O(n) solution to the PnP problem. Int J Comput Vis. 2009;81(2):155.
32. Park K, Patten T, Vincze M. Pix2Pose: Pixel-wise coordinate regression of objects for 6D pose estimation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision; 2019. p. 7668-77.
33. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial nets. Adv Neural Inf Proces Syst. 2014;27.
34. Peng S, Liu Y, Huang Q, Zhou X, Bao H. PVNet: pixel-wise voting network for 6DoF pose estimation . In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2019. p. 4561-70.
35. Wen B, Mitash C, Ren B, Bekris KE. se(3)-TrackNet: data-driven 6D pose tracking by calibrating image residuals in synthetic domains. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE; 2020. p. 10367-73.
36. Li Y, Wang G, Ji X, Xiang Y, Fox D. DeepIM: deep iterative matching for 6D pose estimation. In: Proceedings of the European Conference on Computer Vision (ECCV); 2018. p. 683-98.
37. Rusinkiewicz S, Levoy M. Efficient variants of the ICP algorithm. In: Proceedings Third International Conference on 3-D Digital Imaging and Modeling. IEEE; 2001. p. 145-52.
38. Qi CR, Su H, Mo K, Guibas LJ. PointNet: deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2017. p. 652-60.
39. Gao G, Lauri M, Wang Y, Hu X, Zhang J, Frintrop S. 6D object pose regression via supervised learning on point clouds. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE; 2020. p. 3643-9.
40. He Y, Huang H, Fan H, Chen Q, Sun J. FFB6D: a full flow bidirectional fusion network for 6D pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2021. p. 3003-13.
41. Chen L, Zhang H, Xiao J, Nie L, Shao J, Liu W, et al. SCA-CNN: spatial and channel-wise attention in convolutional networks for image captioning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2017. p. 5659-67.
42. Stevšič S, Hilliges O. Spatial attention improves iterative 6D object pose estimation. In: 2020 International Conference on 3D Vision (3DV). IEEE; 2020. p. 1070-8.
43. Yuan H, Veltkamp RC. 6D object pose estimation with color/geometry attention fusion. In: 2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV). IEEE; 2020. p. 529-35.
44. Zhao H, Shi J, Qi X, Wang X, Jia J. Pyramid Scene Parsing Network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2017. p. 2881-90.
45. Xu D, Anguelov D, Jain A. PointFusion: deep sensor fusion for 3D bounding box estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2018. p. 244-53.
46. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2016. p. 770-8.