Robotics and Autonomous Systems 152 (2022) 104041

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Robotic manipulation based on 3-D visual servoing and deep neural networks

Abdulrahman Al-Shanoon *, Haoxiang Lang *

Department of Automotive and Mechatronics Engineering, Faculty of Engineering and Applied Science, Ontario Tech University, Oshawa, ON, Canada

ARTICLE INFO

Article history:
Received 25 August 2020
Received in revised form 18 December 2021
Accepted 21 January 2022
Available online 3 February 2022

Keywords:
Robot–object-interaction
Object-detection and pose estimation
Deep-learning methods
3D visual-servoing

ABSTRACT

The critical challenge, for robot–object-interaction, is to estimate visually the pose of the target object in a 3D space and combine it into a vision-based control scheme in manipulation applications. This paper proposes a novel reliable framework for deep ConvNet combined with visual servoing using a single RGB camera. We introduce an extensive system called Deep-Visual-Servoing (DVS) that addresses an integration of: (I) training of deep-CNNs using synthetic dataset only and operates successfully in real-world scenario, (II) continuous 3 D object pose estimation as the sensing feedback in a 3D visual servoing control scheme, and (III) design, integration and experimentation of visual servoing approach based on Lyapunov's theory. The proposed deep based learning approach, the kinematic modeling and controller design are experimentally verified and discussed using the 6 DOF UR5 manipulator.

1. Introduction

In order to operate robots efficiently in an unknown and dynamic environment, robots must perceive their surroundings and have enough knowledge about target objects in the real-world. The main challenge of vision-based robot–object-interaction is to estimate the 6 Degrees of Freedom (DOF) pose of the target object. Several studies employ fiducial marker as a point of reference for target object's pose in the workspace. Other studies applied traditional computer-vision methods to understand the pose of objects. For instance, model-based and feature-based methods were used for robot–object-interaction tasks. However, besides the empirical issues that lead to non-practical results, many drawbacks accompany those techniques, such as limited to structured background, poor performance with light and occlusion variations and required highly textured objects.

Deep neural network, specifically Convolutional Neural Network (CNN), has become a promising method for handling object recognition issues. Traditional deep-learning methods have been successfully applied into 2D object detection problems. In addition, several researchers have recently been utilizing traditional deep-learning to achieve 3D object detection and pose estimation. However, unlike 2D object detection, labeling 3D object is difficult and required experts. Using synthetic data for deep neural network training has solved this issue by proposing an unlimited amount of useful pre-labeled training dataset that is produced safely in a reasonable effort.

Contemporary Visual Servoing (VS) studies underline the significance of considering the photometric details of the whole input image into VS system. This introduces such methods called direct/photometric VS that were proposed in [1,2]. However, a range of constraints on such approaches that preclude robust efficiency and other potential improvements. The direct VS methods are strongly affected by the image perturbations (such as pixel intensities, or occlusions) which cause instabilities. The bottleneck could be the different resolutions that highly impact the nature of the available information, various image quality (with various types of noises), or illumination changes that affect the pixel intensities. The main drawback here is the non-reliable convergence domain that happens due to the non-linearities of the cost function to be minimized. In addition, such method is restricted to the empirical setup, VS system only performs eye-in-hand configuration to reposition from starting to desired location. This makes it less feasible and limited to practical scenarios. Since the direct VS methods servo on the entire image features, extra fine-tuning is required to specify to certain location. In case of applying further developments by generalizing on to several target objects with various lightings and backgrounds, direct VS would not be the best choice due to the requirements of re-training datasets and task-specific fine-tuning.

The photometric information of the entire input image is not required in the Position Based VS (PBVS) methods, but the geometrical pose of target object is needed. Recent works [3–7] have reported vision-based robot–object-interaction task without relying on the whole input image. However, none of them has

* Corresponding authors.
E-mail addresses: abdulrahman.alshanoon@ontariotechu.net (A. Al-Shanoon), Haoxiang.Lang@uoit.ca (H. Lang).

https://doi.org/10.1016/j.robot.2022.104041
0921-8890/

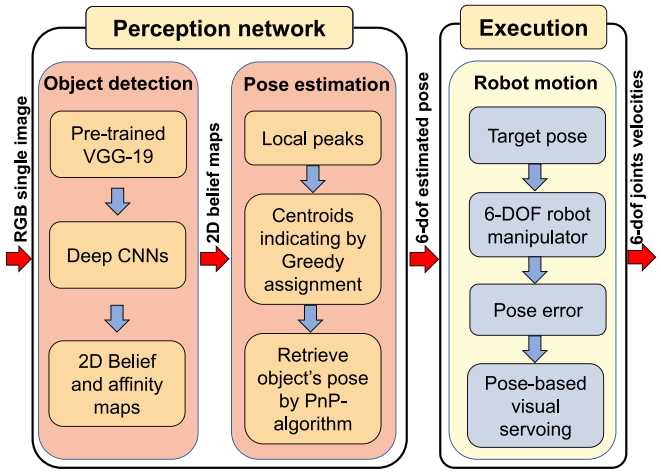

Fig. 1. The proposed architecture of the Deep-Visual-Servoing (DVS) for robotic manipulation systems.

demonstrated a complete framework for achieving 6 DOF pose-based visual servoing from deep-ConvNet. Our contribution in this paper proposes a deep-ConvNet based learning framework for 6 DOF visual servoing, named Deep-Visual-Servoing (DVS). DVS learns from only synthetic dataset and successfully generalizes on real scenario using single RGB image. DVS infers the position and orientation of the target object and achieves 6 DOF pose-based visual servoing task. The proposed pipeline is shown in Fig. 1. There are two main phases explained in the architecture: (1) perception network based on deep-learning to estimate the pose of the object; (2) execution, which achieves the desired pose between the end-effector and the object by utilizing the estimated pose and visual servoing control. The main idea is to utilize CNN to identify the 2D target object in the single RGB input image, followed by pose estimation methods to retrieve the 3D translation and 3D orientation of the target object. The results of 6 DOF object pose information will be utilized as a feedback into vision-based control system so-called pose-based visual servoing. Since the aim of the DVS is to achieve robotic-visual-servoing task of household objects, we evaluate our system based on YCB images [8].

The remainder of the paper is organized as follows: Section 2 introduces the review of background and literature. Section 3 illustrates the modeling of the perception network for 2D object detection and pose estimation, camera, manipulator and visual servoing model development. Section 4 shows experimentation results with discussions. Lastly, the conclusions are set out in Section 5.

2. Related work

This section reviews the previous and existing studies conducted on object detection and 6-dimensional pose estimation from classical techniques to trainable machine-learning methods. Meanwhile, vision-based robot motion control methods are also reviewed.

Traditional approaches utilized RGB images with local interest-points and feature matching algorithms in order to achieve object recognition and pose estimation task [9]. The methods required certain local descriptor for different scale, rotation, and view-points. The drawbacks of such approaches are needed highly textured objects and provide low performance in light changing conditions. Utilizing RGB-D images [10] is another approach to achieve object pose estimation by introducing depth information. This approach has been applied for indoor robotic tasks for grasping applications. However, complex structure and limited background were the issues that hinder the performance.

In recent years, researchers have considered CNN-based methods as the most promising approach for object recognition and pose estimation [5]. PoseCNN suggests regressing 6D pose directly from the image by applying several CNN stages [7]. Other studies documented decent performance and focused on challenges such as occlusions and various light conditions. However, the key matter of training data is how to generate an effective dataset that presents enough variations, where the deep net learns from variety of lighting and poses conditions. Labeling bounding boxes for 2D object detection is simple and effortless to annotate. However, 3D object detection requires high skills and it is impossible to generate labeled training data manually. Some existing labeling tools such as LabelFusion [11] provides useful functions for labeling 3D objects. However, to our knowledge, there is no simple and effective tool which can help to generate real-training-data for 6 DOF object pose estimation that covers enough variations in terms of object poses and lighting conditions. Due to this difficulty, such methods consider real data for training purposes which are highly similar to the test data (e.g. same camera-parameters, object category, and similar lighting conditions). As a result, test data is always limited to any significant difference from the training data.

J. Tobin, R. Fong, et al. in [12] solved this issue by generating the data synthetically which can produce almost unlimited pre-labeled 3D training-data with little effort. A challenge of the current approach to synthetic data is called reality-gap. When the system trains on synthetic data, real data performance would always require extra fine tuning. J. Tremblay, et al. in [13] conducted a recent solution to this issue, where synthetic dataset is randomized in non-realistic methods, in which the test data should seem merely another variance. The solution is called domain randomization which is a combination of non-photorealistic and photorealistic data. The domain randomization of synthetic data yields enough diversity for training dataset that covers variations in poses, lighting, and occlusion conditions. Thus, deep net is capable to perform well on real data with different light conditions without requiring extra fine-tuning. This combination is the state-of-the-art provided by FAT dataset [13] developed by Nvidia research-group and generated based on the YCB household objects. The datasets are generated by the developed tool used for UnrealEngine4 so-called NDDS. Deep-learning methods require a vast number of datasets for training. Therefore, synthetic datasets were provided for training the deep network [14,15]. In this paper, our network is trained on only synthetic dataset that was proved in [13] as the state-of-the-art, which also covers sufficient possibilities of various poses in different environments, for instance, extreme light conditions.

In the VS literature, VS techniques are identified by three distinct classes. Position-based visual servoing (PBVS) methods require the geometric model of target object in order to recover the object's pose with respect to the camera frame. Several studies have documented practical results of considering the 3D position of the targeted object [16,17]. PBVS always relates to the servo error in Cartesian space and it shows a global asymptotically stable system only when the system offers perfection in pose estimation. Previous studies [18,19] have begun to examine the use of all the translations and z-axis into PBVS Cartesian space. However, the other rotations are required to be introduced in 2D image space, which leads to creating a complicated system design. Image-based visual servoing (IBVS) is the approach that does not require a pose estimation. It uses the image features directly since the pose is computed implicitly [20]. It considers the vector which is a set of parameters that are present in image

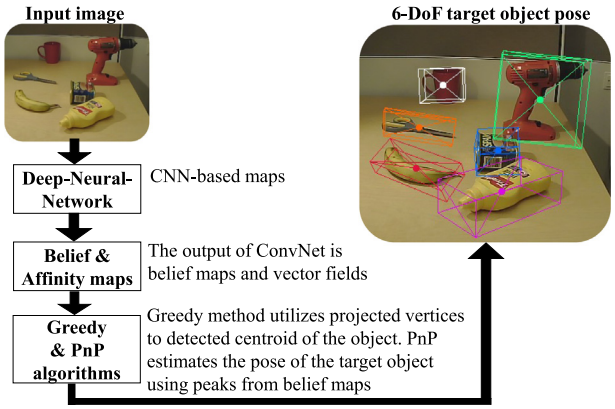

Fig. 2. Object identification and pose estimation of the multiple objects.

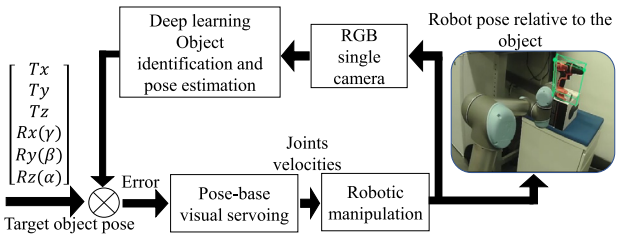

Fig. 3. Deep-Visual-Servoing control diagram.

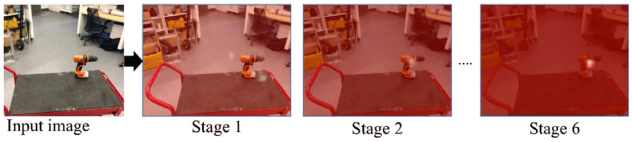

Fig. 4. An input image proceeded in multiple stages (from stage 1 to stage 6) with one vertex.

data. The main control loop architectures of both approaches are relatively similar; the only difference is the input type and the feedback loop. The input parameter in PBVS is the 3D position of an interesting point in the workspace. In [17], two methods of VS were tested to compare their stability and robustness if modeling errors were present in the computation. The authors concluded that both methods were asymptotically stable and locally robust. The third VS class is the hybrid system that uses both 2D and 3D data as visual feedback. This model is a sophisticated implementation since it uses the advantages of both previous approaches [21]. Hybrid robot visual servoing typically utilizes the 2D and 3D target object features to decouple the translational and rotational robot motion.

Advancements in camera technology and processing capabilities have allowed VS to become more effective for real-time operation due to the increase in computational speeds. The effectiveness of VS is seen in its implementation into vision-guided manipulator based on classical computer-vision or traditional deep-machine learning methods. Recently, comprehensive studies were found in the literature on deep-learning methods for object detection and pose estimation [3,4]. In addition, published researches on robot motion control based on vision feedback were held separately. Therefore, there is a need to develop an extensive framework that combines deep-learning approach (trained on the synthetic dataset) for object detection and 6 DOF pose estimation with 6 DOF visual servoing task. In this paper, the proposed framework includes a deep-ConvNet, which learns from computer generated-images with high variations in textures and lighting and operates on real-world scenarios with pose-based visual servoing.

3. modeling and visual servoing control

Our proposed system consists of three main steps. Firstly, deep-ConvNet identifies 2D objects using single RGB image. In the process of training a synthetic dataset was used (only RGB images). It covers varieties of lighting, occlusions, and background conditions. The results yield information of 2D belief maps that represents multiple 2D objects in the image. Secondly, pose estimation algorithms utilize belief maps to retrieve the 3D translational and 3D orientational pose of the target object without post-alignment step, as shown in Fig. 2. Lastly, the desired pose of the target object is directly sent, without the need to post-refinements of the estimated pose, to the visual servoing controller which is in charge of achieving robot pose with respect to the camera frame.

Fig. 3 illustrates the generic diagram of the overall proposed system, where the deep-ConvNet constantly detects and estimates the current object's pose. The generated pose error stimulates the control law to send joints velocities and reduce the error by achieving robot pose relative to the target object.

In this section, the modeling of the three previous steps is demonstrated, including a single-camera model, kinematics of 6DOF manipulator and development of the control law based on Lyapunov's concept.

3.1. Perception network

The proposed perception network consists of two main steps namely: CNN-based and pose-estimation step. The network infers the location of the object with pose estimation in the workspace. Inspired by Convolutional Pose Machines (CPMs) [22], our deep-ConvNet infers the target key-points in a single-shot, fully convoluted deep network and through multi-stage architecture. Each stage of the convolutional neural network generates 2D belief maps of the target object. The input of the subsequent stage takes advantage of both, image features and belief maps of the previous stage. These stages are the sequence of predictors that make better predictions and an increasingly refined estimate of the location of the object. Since the design of the network is convolutional, the early stages (of generating 2D belief map) might have ambiguities that will be resolved by the later stages, as shown in Fig. 4.

The training network utilizes a larger receptive field on both the image features and belief maps, this manner improves the network accuracy. During the training, transform-learning is applied as a feature extraction step, followed by multiple-stages architecture. The final goal of the feedforward network is to detect 2D key-points of the observed objects in the image. The network outputs two types of maps, which are belief map and vector fields, indicating multiple objects of the same category. Training network generates 9 belief maps in each stage, each one represents a vertex which will be at the end 8 projected vertices of the 3D bounding box, and one belief map for the centroids. In the same manner, each stage produces 8 vector fields that indicate the direction to the centroid from each 8 projected vertices. To indicate the object's centroid, the training network always seeks for the local peaks from the belief maps. It uses greedy algorithm to associate the projected vertices to the indicated centroids, similar to [4,5]. Greedy algorithm assists finalizing the object's centroid, in addition to determine multiple

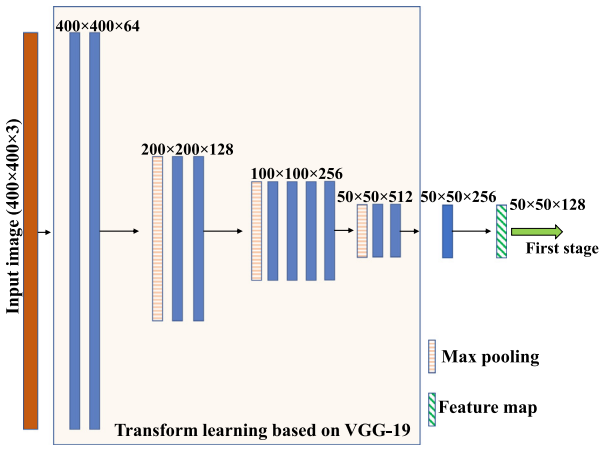

Fig. 5. Network architecture based on VGG-19.

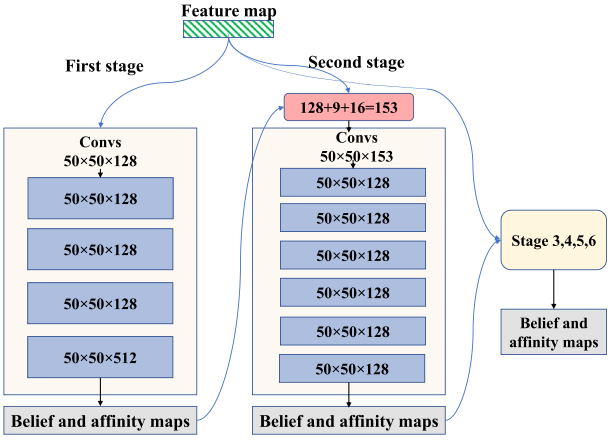

Fig. 6. Proposed CNN-based architecture for 2D belief maps.

centroids for multiple target objects observed in the image. Then, the object's projected vertices of the 3D bounding box are utilized by Perspective-n-Point (PnP) algorithm to retrieve the 6 DOF object's pose [23]. PnP algorithm at least requires four vertices to obtain the pose. To demonstrate robust performance, nine points of the generated vertices are used to solve the PnP approach. Intrinsic camera parameters and object dimensions are needed to estimate the translation and orientation of the target object relative to the camera frame.

In Fig. 5, the network trains directly on the input RGB image ($400 \times 400 \times 3$) and the first 10 layers of the network compute the image features based on VGG-19 model [24], pre-trained on ImageNet [25]. Then, features dimensions minimized from 512 to 256 and from 256 to 128 by using two 3×3 convolutional neural layers.

The feature map dimension-128 is the input to the first stage of training that contains 3 layers of ($50 \times 50 \times 128$) and one layer of ($50 \times 50 \times 512$), as shown in Fig. 6. This stage followed by belief map ($50 \times 50 \times 9$) and vector field ($50 \times 50 \times 16$) that both are fed to the next stage, including the output of the feature map. Similarly, the remaining training stages (from 2 to 6) should have same structure as the first stage. The receiving dimension input ($128+9+16 = 153$) is an output of image features map, as well as the belief map and vector field of the immediately preceding stage. In the remaining stages, there are 6 layers of ($50 \times 50 \times 128$) and the output is belief map and vector field. The output of a certain layer maps to the subsequent input layer

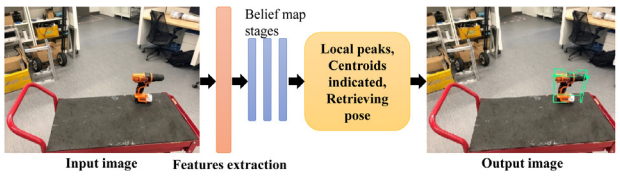

Fig. 7. Object pose estimation in unprepared environment.

by Rectified Linear Unit (ReLU) activation functions which always keep the positive value.

For the training dataset, more than 60k of synthetic images were considered based on YCB object-dataset. (RGB images only, segmentation process and depth images are not required.) The dataset images used for training session are entirely computer-generated datasets which carry information about target object presents in the image. For example, information about location, centroid, and object's category. The ground truth information is the coordinates of the bounding box of objects existed in the training image. This ground truth information will eventually be compared with the projected vertices produced by the proposed ConvNet model.

The image features extractor for first layers-set learns from VGG-19 pre-trained model. The system has been implemented on PyTorch platform [26] and trained on 8 GPUs (each one is Nvidia Tesla K-80) for 70 epochs with a batchsize of 128, the results have been tested on CPU Intel. Core i7-7700k. Learning rate is 0.0001 based on network's optimizer, similar to Adam [27]. During the training, the regularization method of the Loss function (L2) is used to calculate the loss error between the predicted output and the true value (ground-truth) for each input. At the end of each epoch, error value is accumulated by applying Mean Square Error (MSE) of each input. This is applied for the entire training phase which shows how the model learns by minimizing the square of the differences between the true and estimated values. Loss function will build and label belief and affinity maps. The used L2 loss function calculates the loss error between the predicted belief maps and the true value (ground-truth) of the training data. The total loss in the stage i is the sum of losses $L = \sum_i^n L_i$, where L_i is defined below:

$$L_i = \frac{1}{n} \sum_{i=1}^{n} \sum_{v=1}^{m} \left(B_v^i - \dot{\mathrm{B}}_v \right)^2 \tag{1}$$

where B_v^i is the CNNs output for a belief map at stage $i \in (1\ldots n)$ for vertex v ϵ $(1\ldots m)$, n is the stage number and m stands for the number of vertices, $\dot{\mathrm{B}}_v$ is the ground truth. This approach called intermediate supervision where the gradient is stored at the end of each stage, this avoids the well-known issue vanishing gradient.

To obtain the 3D bounding box, the 2D object vertices output from belief maps followed by PnP iterative approach, as shown in Fig. 7.

The purpose of PnP problem is to estimate the translation and orientation of the calibrated camera from the known 3D points (the object dimensions) to the corresponding 2D image projections. This method helps to find the object pose from 3D-2D correspondences point, which is based on Levenberg–Marquardt optimization approach, where the proper object pose should be found by minimizing the reprojection error (RE). RE is the sum of the squared distances between the observed 2D projections image-points and projected 3D object-points, as shown below.

$$x^2\,(m, b) = \Delta y_1^2 + \Delta y_2^2 + \Delta y_3^2 + \cdots \tag{2}$$

The PnP input is the intrinsic camera parameters, target object dimensions, and 2D observed points. The output results retrieve

the rotational and translational vectors of the 3D object into 2D image plane. There are three coordinate frames namely: world, camera, and the image plane. If the orientation and translation of a 3D point are known in the world coordinate, the corresponding point could be transformed into camera coordinate, using Eq. (3). Then, the 3D point can be projected into the image plane, by utilizing the camera intrinsic parameters. Let P_i^w be the 3D point in the workspace coordinates, which is shown in Eq. (3) relative to C_i^C camera coordinate system. The 3D point $\begin{bmatrix} x_i^w & y_i^w & z_i^w \end{bmatrix}^T$ is projected into image plane m_i as $\begin{bmatrix} u_i & v_i \end{bmatrix}^T$. The perspective transformation for the pinhole camera model is shown in Eq. (4).

$$C_i^C = \mathrm{R}\, P_i^w + t \tag{3}$$

$$\begin{bmatrix} x_i^c \\ y_i^c \\ z_i^c \end{bmatrix} = \mathrm{R} \begin{bmatrix} x_i^w \\ y_i^w \\ z_i^w \\ 1 \end{bmatrix} + t$$

$$Sm_i = K[\mathrm{R}|\mathrm{t}]P_i^w \tag{4}$$

$$\mathrm{S}\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_r \\ 0 & f_y & c_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x_i^w \\ y_i^w \\ z_i^w \\ 1 \end{bmatrix}$$

Where K is the intrinsic camera matrix, [R|t] is the extrinsic joint matrix which represents rotation and translation vectors respectively. It is used to describe the homogeneous rigid motion of the object point with respect to the camera coordinates and translates the coordinates of each 3D point into the coordinate system relative to the camera frame. S denotes a scalar projective factor. f_x and f_y are the focal length expressed in pixel coordinates. c_r and c_c are the principal point that is image center in pixel frame.

If P_i^w is known and the camera intrinsic is also known, therefore, the projected image point m_i could be determined. However, [R|t] is unknown, PnP algorithm is used for non-linear system to determine such a pose that minimizes RE, an approximated estimation of [R|t] could be obtained by iteratively changing the estimated [R|t] and reducing RE. The process requires the object dimensions and 2D observed projection points proposed by deep-ConvNet and vertices.

3.2. Single camera and manipulator modeling

Single Camera. Investigating the geometry of the image formation process can demonstrate the relationship between the interest feature points in the world frame and their projections on the image plane. A common image formation model is the Pinhole lens approximation [28]. As shown in Fig. 8(a), P is the object point in the workspace with coordinate $P = (x, y, z)$.

The point P is projected as $p\ (u, v)$ on the image plane with coordinate (u, v, f). The perspective projection of the point P in the camera coordinate frame is (x_c, y_c, z_c), and the same point is represented as (r, c) into pixel coordinate frame. The intersection of the z-axis with the image plane is the principal point (c_r, c_c), which is measured with respect to the coordinates of the pixel frame. f is the distance between the image plane and camera frame (camera focal length). In the case where f is known and the coordinate of the point P in the camera frame (x_c, y_c, z_c) is also known. Then, the 3D point of P in the workspace can be determined by indicating the projected point (u, v) in the image plane [29]. The perspective projections result in Eq. (5).

$$u = \frac{fx}{z}, \quad v = \frac{fy}{z} \tag{5}$$

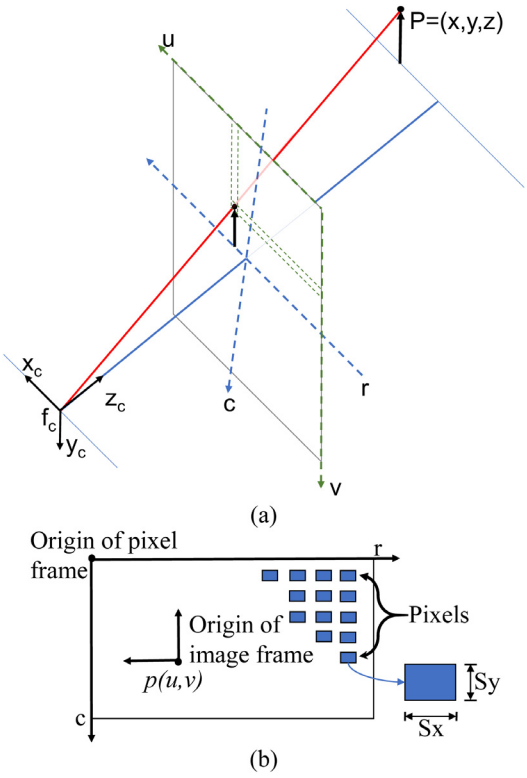

Fig. 8. Single camera model, (a) pinhole model, (b) pixel coordinate frame.

This helps to find the object position in the workspace by introducing (u, v). However, the sensor of the digital camera (CCD/CMOS), which is a 2D array, is measured in pixels. This is shown in Fig. 8(b) where $(u, (v)$ is the origin of the image plane and (r, c) is the origin of the pixel frame. The pixel shapes are usually rectangular, and the pixel's width and height are given as Sx and Sy respectively. The centers of the image plane and pixel coordinate frame are not the same. Therefore, the coordinate transformation between the image plane and the pixel frame are expressed below:

$$r = -f_x \frac{X_c}{Z_c} + c_r, \quad c = -f_y \frac{Y_c}{Z_c} + c_c \tag{6}$$

In order to obtain the position of the projected object from the world into the image plane, pixels are used as shown in the previous equations. However, the various camera parameters such as Sx, Sy, u, v, camera focal length (fx, fy), and (c_r, c_c) are unknown. These parameters are known as intrinsic camera parameters which could be obtained through camera calibration and they are constant for a given camera. A chess board of known dimensions and number of squares was availed to calibrate the used camera and obtain the necessary parameters.

Robot manipulator. Fig. 9 demonstrates the kinematics frames of 6DOF manipulator when the joints angles are zero. Homogeneous representation is used to define the relationship between the frames of the manipulator.

Fig. 10 demonstrates the primary frames of interest used in our experiments namely, robot-based frame, end-effector frame, and camera frame. The relationship between the robot-base frame and the camera-frame is fixed. Thus, the skew symmetric matrix $(s(d))$ is required to define the location of the camera relative to the manipulator's base frame. These parameters are in terms of physical measurements from the robot.

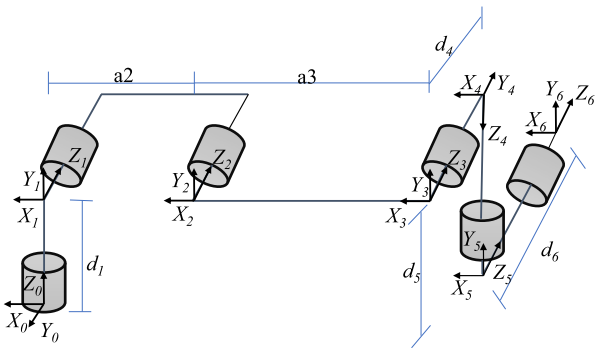

Fig. 9. Manipulator at zero position.

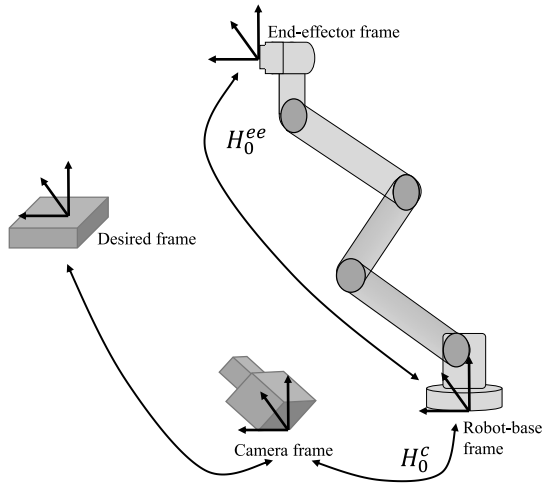

Fig. 10. Frames of interest.

The joints velocities expressed below in to end-effector velocity with respect to itself (ξ_{ee}^{ee}):

$$\xi_{ee}^{ee} = E^{-1} J \begin{bmatrix} \theta_1^{\cdot} & \dots & \theta_n^{\cdot} \end{bmatrix}^T \tag{7}$$

where J is the Jacobian matrix. This equation will be used in the next section to complete the control law with generated errors.

3.3. System modeling and controller design

The 3D Cartesian control task is to compute the desired pose of the robot that eliminates the error measurement comparing with the current pose [30]. Eq. (8) explains the PBVS parameters:

$$\mathrm{S} = (\mathrm{T}, \Theta\mathrm{U}) \tag{8}$$

where T stands for translational vector and ΘU denotes as a rotational vector. Eq. (9) shows the error dynamic task.

$$\mathrm{e} = (\mathrm{T}_e^{ed} - \mathrm{T}_e^{ee^*_d}, \Theta\mathrm{U}) \tag{9}$$

Where S expresses as $\mathrm{S} = (\mathrm{T}_e^{ed}, \Theta\mathrm{U})$, as a current end-effector pose (relative to the desired pose) and could be $\mathrm{S} = (\mathrm{T}_e^{ee^*_d}, \Theta\mathrm{U})$ as a desired end-effector pose.

The overall control scheme is derived from both the position and orientation vectors introduced earlier. The change in position vector can be expressed in terms of a rotation matrix between the current and desired end-effector pose, R_e^{ed}, and the end-effector velocity as seen in Eq. (10). Isolating for the end-effector velocity yields (11).

$$\dot{T_e^{ed}} = R_e^{ed} T_e^{ed} \tag{10}$$

$$T_e^{ed} = \left(R_e^{ed}\right)^T \dot{T_c^{ed}} \tag{11}$$

Repeating the general process for the orientation vector begins with Eqs. (12) and (13).

$$\dot{w_e^{ed}} = R_e^{ed} w_e^{ed} \tag{12}$$

$$w_e^{ed} = \left(R_e^{ed}\right)^T \dot{w_e^{ed}} \tag{13}$$

Angular velocities are determined by the rate of change in Euler angles found in $\phi_e^{ee_d}$ which are transformed using $T(\phi)$ defined in Eq. (14).

$$\dot{w_e^{ed}} = T(\phi)\, \dot{\phi_e^{ee_d}} \tag{14}$$

Where $T(\phi) = \begin{bmatrix} 0 & -sin\varphi & cos\varphi * cos\theta \\ 0 & cos\varphi & sin\varphi * cos\theta \\ 1 & 0 & -sin\theta \end{bmatrix}$.

Substituting (14) into (13) determines the velocity of the end-effector in terms of angular velocities.

$$w_e^{ed} = \left(R_e^{ed}\right)^T T(\phi)\, \dot{\phi_e^{ee_d}} \tag{15}$$

PBVS control seeks to eliminate the error pose between the desired and the current end-effector pose by controlling the motion of the robot and sending the required translational and rotational commands to a manipulator. The basis of the control law begins with Lyapunov's proportional control scheme, where k is the proportional gain and the solution yields an exponential decrease of error described below:

$$\dot{e}(t) = -ke(t) \tag{16}$$

Visual servoing error is defined by the difference between the current image and camera parameters, s, and the desired image and camera parameters, s_d.

$$e(t) = s - s_d \tag{17}$$

The robotic system used in this research operates in three-dimensional Cartesian space therefore the parameters in Eq. (17) must be defined by two vectors representing the position and orientation of the end-effector. This data is acquired by pose-estimation algorithms that use a 3D target object as a reference to the image data to determine the position and orientation of the end-effector. Eq. (18) defines the results of the pose estimation algorithm.

$$s = \left(T_e^{e_d}, \phi_e^{ee_d}\right), s_d = (0, 0) \tag{18}$$

Where $T_e^{e_d}$ is the position vector of the current end-effector frame with respect to the desired end-effector frame. $\phi_e^{ee_d}$ is the orientation vector of the current end-effector frame with respect to the desired frame described by Euler angles. The desired pose vector is zeros. Substituting these vectors into (17) yields the following error equation.

$$e(t) = \left(T_e^{e_d}, \phi_e^{ee_d}\right) \tag{19}$$

Substituting Eq. (10) into (19) can produce the following equation.

$$T_e^{ed} = \left(R_e^{ed}\right)^T \dot{T_c^{ed}} = \left(R_e^{ed}\right)^T \dot{e(t)}_t \tag{20}$$

Similar to the translational error, the rate of change in orientation error is defined by $\dot{\phi_e^{ee_d}}$, which can be substituted into (19) to generate the equation below:

$$w_e^{ed} = \left(R_e^{ed}\right)^T T(\phi) * \dot{e(t)}_W \tag{21}$$

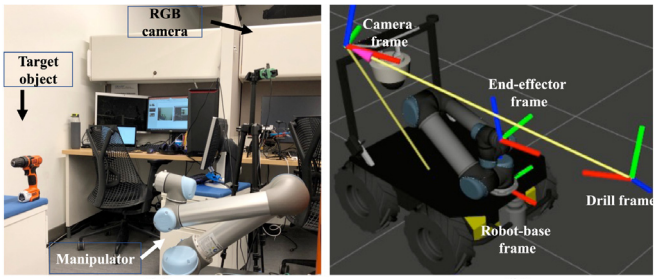

Fig. 11. (a) Experimental setup; (b) ROS 3D visualizer (Rviz).

Table 1
Speed network performance for different stages.

Number of processing stages	Time of processing (sec.) up to n-stage.
1-stage	0.01961708068
2-stage	0.04075098037
3-stage	0.06070590019
4-stage	0.07920598983
5-stage	0.09965205192
6-stage	0.1820244

The control law for the end-effector velocity is obtained by applying a proportional control scheme of Lyapunov from Eqs. (16) to (20) and (21).

$$\xi_{ee}^{ee} = \begin{bmatrix} T_e^{ed} \\ w_e^{ed} \end{bmatrix} = -k \begin{bmatrix} \left(R_e^{e_d}\right)^T & 0 \\ 0 & \left(R_c^{e_d}\right)^T * T\left(\phi\right) \end{bmatrix} e(t) \tag{22}$$

Eq. (22) can be simplified to $\xi_{ee}^{ee} = -kLe(t)$, where L is known as the Interaction Matrix. The derived control law controls the joint velocities in proportion to the error between the actual and desired end-effector robot pose. By combining this with the previous system model developed in Eq. (7), the complete control law is designed in Eq. (23). In order to analyze the performance characteristics of the robot model, this control is implemented physically by experimentations in the following section.

$$\begin{bmatrix} \theta_1 & \dots & \theta_n \end{bmatrix}^T = -k\, L\, E\, J^{\dagger}\, e(t) \tag{23}$$

Where $J^{\dagger}$ is the pseudoinverse method of Jacobian matrix that used for solving inverse kinematics with J.

4. Experiments and results

Fig. 11(a) shows the physical implementations of the DVS system and the testing setup that were occurred in unprepared lab environment. In terms of camera installations, a single RGB camera was used to test two configurations. An eye-in-hand where target object and camera are both moving simultaneously, and eye-to-hand setup where the camera can observe both the target object and the robot. The physical setup is also represented in the Robot Operating System (ROS) framework to accomplish an autonomous operation. Fig. 11(b) demonstrates the frames of interest, namely robot-base, camera, end-effector, and target-object frame, all are exhibited in ROS 3D visualization tool called Rviz.

For testing procedures, six different objects (drill, mug, scissors, banana, meat can, and mustard) were arbitrarily chosen as target objects during the experiments. We leverage the image-data from YCB household objects. Different virtual environments used as object-backgrounds to create photorealistic synthetic dataset, only RGB images were utilized for training phase, depth images and segmentations are not required. As a first place of DVS and proof-of-concept, the selected target objects suffice to hold desired features that are necessary for object detection and pose estimation algorithms.

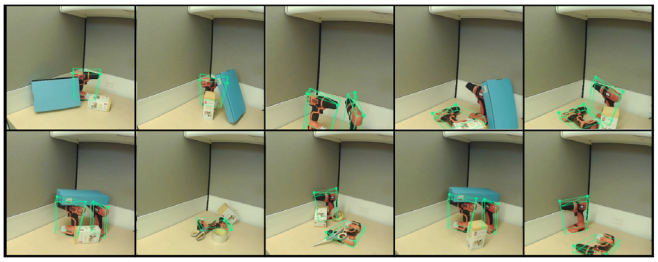

Fig. 12. Pose estimation of target-objects with occlusion.

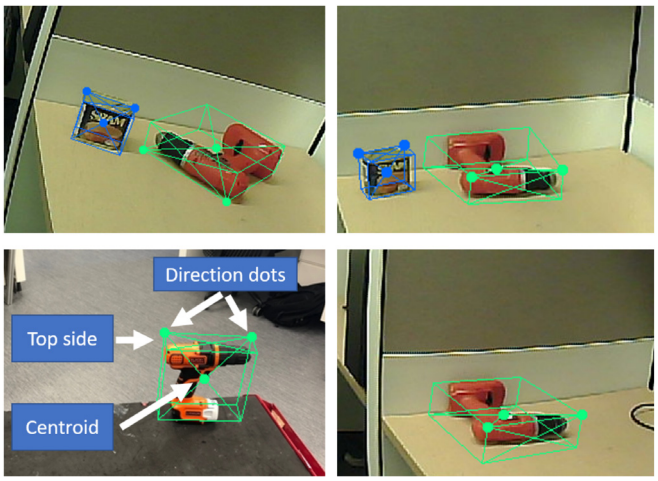

Fig. 13. Characteristics of the 3D bounding box.

Deep ConvNet 6D object pose estimation. The proposed perception network has tested target objects in multiple poses including different occlusion situations. The experiments findings estimated the pose successfully, even in case of occlusion of the object is happening in camera scenes. Fig. 12 demonstrates pose estimation of multiple target objects (of same category) in different poses. The ultimate tests showed a sufficient model accuracy which can achieve 6 DOF robot visual servoing task.

Fig. 13 demonstrates the characteristics of the 3D bounding box which includes the target object. The top side of the bounding box is represented with (X) shape. The two dots on the top side indicate the direction of the target object. Finally, the object centroid is shown as a dot, as seen by the white arrow in the figure.

Table 1 documented the execution time of object detection and pose estimation for different number of stages using a Drill as a target object. The processing time for the sixth stage includes all of the preceding stages as well as the final one.

Table 2 illustrates the processing time to find 2D projected points (CNN-based), retrieving target pose, and total perception network time for 6 different target objects. Total processing time varies from 180 ms to 250 ms to detect and estimate 6 DOF pose of target object.

The output of the series of the CNNs indicates 9 vertices of belief maps. In order to extract individual objects from the belief maps, greedy algorithm determines the centroids of the multiple target objects. In addition, affinity map is utilized to find multiple instances of the same object in the image. Belief and affinity maps show how the model improves throughout multiple stages, Fig. 14(a) and (b) illustrate the difference between the

Table 2
Processing time of the perception network, including 2D estimated key-points and retrieving target pose for six different target objects.

Target objects	Processing time (sec.) to find 2D key-points CNN-based.	Processing time (sec.) to retrieve target object pose.	Network elapsed time (sec.) after 6 stages.
Drill	0.0032057	0.00031208	0.18202
Banana	0.0045201	0.0003559	0.25114
Scissors	0.004116	0.0004518	0.25149
Mug	0.004416	0.000319	0.27156
Mustard	0.004464	0.00033807	0.25168
Meat can	0.004755	0.00033497	0.25134

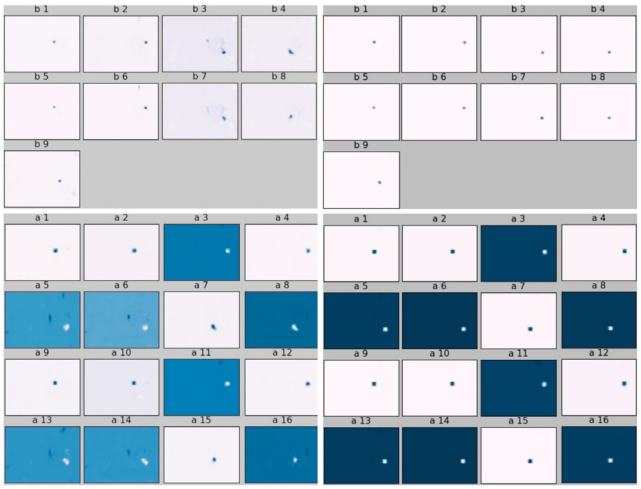

Fig. 14. Belief maps (top) affinity maps (bottom) for: (a) stage 1; (b) stage 6.

Fig. 15. Nine vertices of the final stage cuboid points, and a centroid vertex.

results of the first and sixth stages respectively. Fig. 14(a) (top and bottom) shows the output of 9 vertices of belief maps and output of affinity maps respectively for stage 1. Fig. 14(b) (top and bottom) shows how the belief maps are improved after 6 stages. As mentioned earlier, increasingly larger receptive fields cover more context and resolve the problem of ambiguity that clearly occurs in the early stages.

As a result of feature extractions, feature map is fed to a series of CNNs that output belief map tensor. Each belief map tensor represents one of each 9 vertices of the 3D bounding boxes. Fig. 15 illustrates the combination of the 9 vertices (of final stage) that can form bounding cuboid as well as one belief map for the centroid. Similarly, multiple perception networks work simultaneously to infer multiple instances of the object, the combined vertices in Fig. 15 are obtained from a single image which is shown in Fig. 4.

Table 3
Average-distance between the projected points and the ground truth for six different objects.

Target object	Average distance (mm)
Drill	0.4622
Mug	0.9779
Banana	0.3263
Mustard	0.4746
Meat can	0.5999
Scissors	2.3628

These vertices are beneficial to be utilized as the current 2D object-points, which are needed through PnP process. The 3D bounding box could be retrieved by knowing the intrinsic parameters from the calibrated camera and object dimensions. Reprojection Error (RE), which starts with a larger value, will be minimized by proposing a more accurate object pose.

For the accuracy performance, evaluation metric was calculated on multiple target objects. Average-distance is the difference between the ground truth and the estimated key-points from the perception network. Findings of the average-distance are reported in Table 3. The accuracy-threshold for the robotic manipulation was measured experimentally to find the necessary level of accuracy for grasping purposes. Accuracy-threshold was found around 15 mm, by calculating the difference of centroids between ground truth and estimated points, using our robotic system (UR5 manipulator and 2-finger gripper from RobotiQ).

During training phase, the datasets cover a wide spectrum of varieties. Included all possible poses of target objects during various of occlusion events and lighting changes. In this case, real-life testing will just appear as another variance. Considering a diverse of training situations (including various backgrounds, lightings, and occlusions) will dramatically improve the real-life performance. In addition, the multi-stage architecture of the perception network greatly assists to resolve the uncertainties and ambiguities that associated with occlusion scenarios. Throughout the stages, predictions are gradually corrected and improved. The deep-ConvNet utilizes intermediate supervision technique to replenish the gradient at the end of each stage. The intermediate representations are significant to generate increasingly accurate belief maps. It provides essential structure to improve generalization abilities. To test the real-life generalization performance of the perception network, Fig. 16 demonstrates the detection and pose estimation findings for six different objects. Including, (top images) different real-world unstructured backgrounds in the unprepared lab environments, (middle images) several occlusion events, (bottom images) various light conditions (extreme and dim lights). A light source was used to disturb the image view and examine the network capabilities during light variations. It is worth mentioning that perception network ultimately performed well without requiring any type of post-refinements.

Deep-based 3D visual-servoing. After estimating the 6 DOF pose of the target object, the execution network operates autonomously to achieve a complete task of deep-based visual servoing. Without the need to post-refinements of the estimated pose of a target object. Two rounds of tests were carried out to show the performance of the entire system. First, a straight test where the target object was placed in front of the robot. The end-effector of the robot moved as expected towards the detected object. Likewise, a second test was performed but deliberately positioned target object at an angle to cause a curved trajectory for the end-effector.

More measurements are extracted during the test experiments (straight and curved) and shown in Fig. 17. The linear and angular end-effector velocities of both experiments are shown at (a) and (b) of both sides of Fig. 17, correspondingly. Similarly, on both

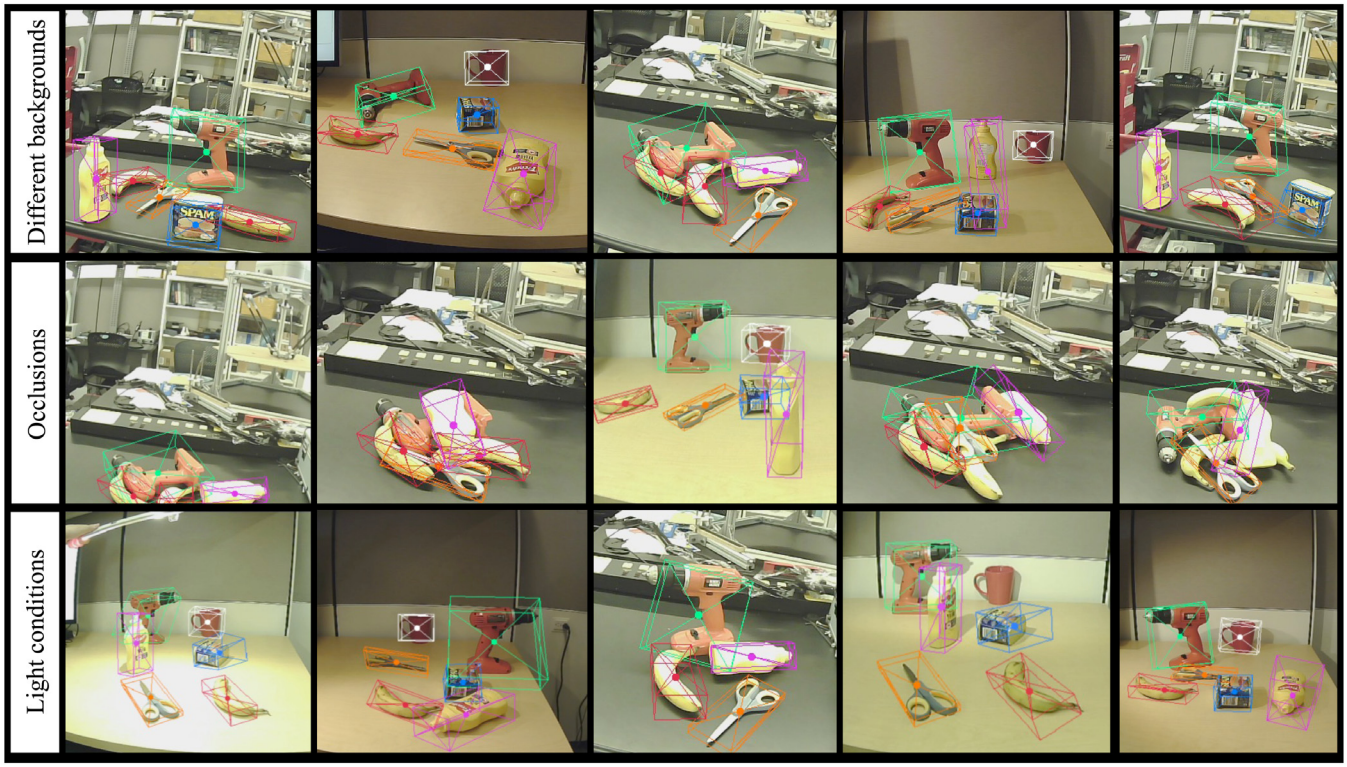

Fig. 16. Perception network performance on six different objects in unstructured various situations. (Top) different backgrounds in unprepared lab environments, (middle) occlusion events, and (bottom) various lighting conditions.

sides, errors of position and orientation are shown in (c) and (d), respectively, representing the measured errors between the end-effector and the target object during the time of operation. The error minimizes when the end-effector moves close to the object and becomes within the user-defined safety distance. The 3D trajectory of the end-effector is drawn in part (e) of both sides of Fig. 17.

Further examination of the real-time tracking test is also required in order to investigate the capability of the proposed system. During the process, the empirical tracking experiment with occlusion shows the efficiency of a continuous visual servoing task. It shows that both perception and execution networks have performed reliable performance and never lose tracking of the desired pose (a video has been included in this submission).

Prior visual servoing studies [3–7,31] requiring (i) complex vision-based setup to obtain pose estimation with limited background structure, (ii) or training on synthetic and real-world dataset collections including fine-tuning efforts. Thus, after training on synthetic datasets, implementing these methods in new real-world environments is difficult and limited. Comparing to such methods, our system estimates object poses competitively, which are trained only on synthetic data and generalized to physical 3D pose-based visual servoing. The use of domain randomized synthetic dataset was (i) applied in the context of 6 DOF object pose estimation from single image (ii) and executed in 3D continuous visual servoing task. Our findings have shown the physical capabilities of generalization to novel environments for the handling of light and occlusion variations.

5. Conclusion

This paper presented a deep-ConvNet based learning framework for 3D visual servoing robotic manipulation, named Deep-Visual-Servoing (DVS). The proposed system detected and estimated the pose of objects in 3D space using an effective deep-CNN model architecture. The pose estimation data was then used in a 3D visual servoing scheme to control the motion of a robotic manipulator (UR5). First, a perception network utilized a single RGB camera as input. Then, CNNs sequence-layers were employed to predict 2D key-points of target objects followed by the indication of the object's centroids and pose in 3D space based on the greedy assignment and PnP technique. The output of the deep ConvNet was fed into 3D visual servoing to control joints' velocities to obtain the desired robot pose with respect to the target object.

Our scalable system overcomes the limitations accompany the direct VS methods. The proposed system entirely trains on synthetic RGB single images which involves multiple poses that cover various of occlusions and light conditions. Extra fine-tuning is not required during generalizing on different real-world backgrounds. To provide scalable VS system, we experimentally validated the task on two different VS configurations, eye-in-hand and eye-to-hand. The proposed VS system is able to robustly operate even when both the camera and target object are simultaneously moving. This assists to avoid uncertainties of the environment and robotic kinematics model.

The accuracy of the entire system is experimentally tested showing the capability of performing robotic manipulation task. The results of deep ConvNet showed that it was sufficient to be trained using synthetic datasets and without post-alignment for robotic manipulation purposes. The proposed system has physically implemented on 6 DOF manipulator to extract the performance characteristics of the robot model based on the feedback from the proposed deep neural network. The findings of experimentation have resulted in a robust and continuous visual servoing operations with handling occlusion and light variations. In terms of future work, it would be interesting to set the system without the need to camera calibration parameters. Further experiments aimed at transparent objects will also be interesting.

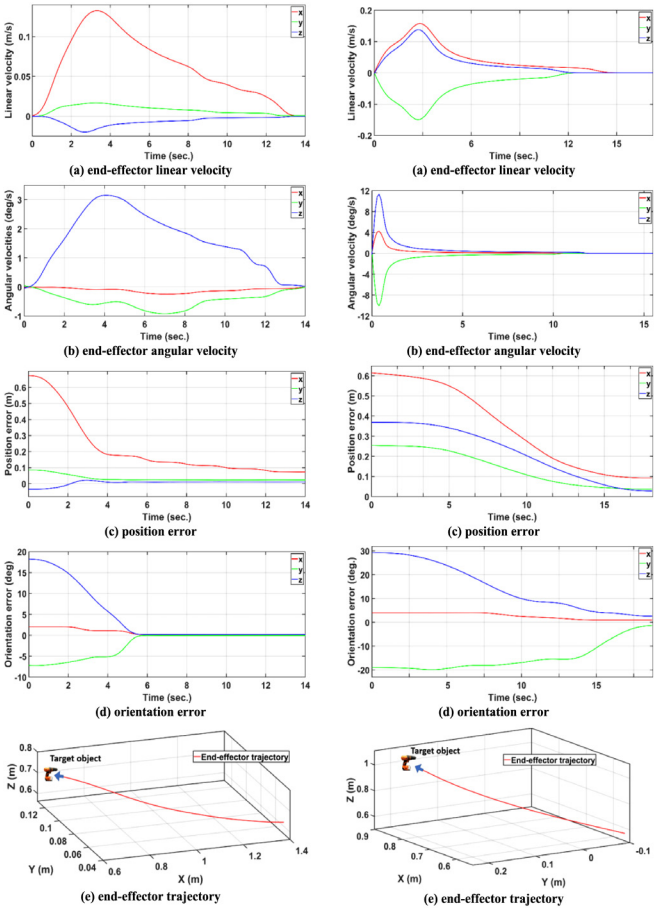

Fig. 17. Results of experiments demonstrate velocities, pose error, and 3D end-effector trajectory during straight (left-side) and curved tests (right-side).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the NSERC Discovery Program under Grant RGPIN-2017-05762 and the MITACS Accelerate Program under Grant IT14727.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2022.104041.

References

[1] Q. Bateux, E. Marchand, J. Leitner, F. Chaumette, P. Corke, Training deep neural networks for visual servoing, in: 2018 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2018, pp. 1–8.

[2] E. Marchand, Subspace-based direct visual servoing, IEEE Robot. Autom. Lett. 4 (3) (2019) 2699–2706.

[3] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, N. Navab, Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1521–1529.

[4] M. Rad, V. Lepetit, 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, pp. 3828–3836.

[5] B. Tekin, S.N. Sinha, P. Fua, Real-time seamless single shot 6d object pose prediction, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 292–301.

[6] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, S. Birchfield, Deep object pose estimation for semantic robotic grasping of household objects, 2018, arXiv preprint arXiv:1809.10790.

[7] Y. Xiang, T. Schmidt, V. Narayanan, D. Fox, Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes, 2017, arXiv preprint arXiv:1711.00199.

[8] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, A.M. Dollar, The ycb object and model set: Towards common benchmarks for manipulation research, in: 2015 International Conference on Advanced Robotics, ICAR, IEEE, 2015, pp. 510–517.

[9] B. Ahn, D.G. Choi, J. Park, I.S. Kweon, Real-time head pose estimation using multi-task deep neural network, Robot. Auton. Syst. 103 (2018) 1–12.

[10] J. Sock, S. Hamidreza Kasaei, L. Seabra Lopes, T.K. Kim, Multi-view 6D object pose estimation and camera motion planning using RGBD images, in: Proceedings of the IEEE International Conference on Computer Vision Workshops, 2017, pp. 2228–2235.

[11] P. Marion, P.R. Florence, L. Manuelli, R. Tedrake, Label fusion: A pipeline for generating ground truth labels for real rgbd data of cluttered scenes, in: 2018 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2018, pp. 1–8.

[12] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, P. Abbeel, Domain randomization for transferring deep neural networks from simulation to the real world, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2017, pp. 23–30.

[13] J. Tremblay, T. To, S. Birchfield, Falling things: A synthetic dataset for 3d object detection and pose estimation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 2038–2041.

[14] R. Figueiredo, A. Dehban, P. Moreno, A. Bernardino, J. Santos-Victor, H. Araújo, A robust and efficient framework for fast cylinder detection, Robot. Auton. Syst. 117 (2019) 17–28.

[15] J. Tremblay, et al., Training deep networks with synthetic data: Bridging the reality gap by domain randomization, in: Proceedings of the IEEE

Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 969–977.
[16] A. Al-Shanoon, A.H. Tan, H. Lang, Y. Wang, Mobile robot regulation with position based visual servoing, in: 2018 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications, CIVEMSA, IEEE, 2018, pp. 1–6.
[17] F. Janabi-Sharifi, L. Deng, W.J. Wilson, Comparison of basic visual servoing methods, IEEE/ASME Trans. Mechatronics 16 (5) (2010) 967–983.
[18] C. Cai, N. Somani, A. Knoll, Orthogonal image features for visual servoing of a 6-DOF manipulator with uncalibrated stereo cameras, IEEE Trans. Robot. 32 (2) (2016) 452–461.
[19] O. Tahri, A.Y. Tamtsia, Y. Mezouar, C. Demonceaux, Visual servoing based on shifted moments, IEEE Trans. Robot. 31 (3) (2015) 798–804.
[20] A. Al-Shanoon, H. Lang, Y. Wang, Vision-based hand gesture recognition with deep machine learning for visual servoing, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, vol. 51814, American Society of Mechanical Engineers, 2018, V05BT07A050.
[21] C.Y. Tsai, C.C. Wong, C.J. Yu, C.C. Liu, T.Y. Liu, A hybrid switched reactive-based visual servo control of 5-DOF robot manipulators for pick-and-place tasks, IEEE Syst. J. 9 (1) (2014) 119–130.
[22] S.E. Wei, V. Ramakrishna, T. Kanade, Y. Sheikh, Convolutional pose machines, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4724–4732.
[23] V. Lepetit, F. Moreno-Noguer, P. Fua, Epnp: An accurate o (n) solution to the pnp problem, Int. J. Comput. Vis. 81 (2) (2009) 155.
[24] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv preprint arXiv:1409.1556.
[25] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, Ieee, 2009, pp. 248–255.
[26] A. Paszke, et al., Automatic differentiation in pytorch, 2017.
[27] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[28] Y. Wang, H. Lang, C. De Silva, A hybrid visual servoing controller for robust manipulation using mobile robots, IEEE/ASME Trans. Mechatronics 15 (5) (2010) 757–769.
[29] M. Arbabmir, M. Ebrahimi, Visual–inertial state estimation with camera and camera–IMU calibration, Robot. Auton. Syst. 120 (2019) 103249.
[30] G. Dong, Z.H. Zhu, Kinematics-based incremental visual servo for robotic capture of non-cooperative target, Robot. Auton. Syst. 112 (2019) 221–228.
[31] G.Z. Gandler, C.H. Ek, M. Björkman, R. Stolkin, Y. Bekiroglu, Object shape estimation and modeling, based on sparse Gaussian process implicit surfaces, combining visual data and tactile exploration, Robot. Auton. Syst. 126 (2020) 103433.

Abdulrahman Al-Shanoon received the B.Eng. eng. in medical instrumentation, Middle Technical University, Iraq, in 2011, and the M.Sc. Faculty of Engineering, Universiti Putra Malaysia, Malaysia, in 2016. He is currently pursuing the Ph.D. program with the Faculty of Engineering and Applied Science, Ontario Tech. University, Ontario, Canada. His current research interests include mechatronics, autonomous robotics, visual servoing, and machine learning.

Haoxiang Lang received the Ph.D. degree from the Department of Mechanical Engineering, The University of British Columbia, Vancouver, BC, Canada, in 2012. Subsequently, he worked as a Postdoctoral Research Fellow in the Industrial Automation Laboratory of the University of British Columbia. Currently, he is an Assistant Professor in the Department of Automotive, Mechanical and Manufacturing Engineering, University of Ontario Institute of Technology. His research and development areas are Mechatronics, autonomous robotics, visual servoing, advanced controls, and machine learning.