IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

A Hyper-network Based End-to-end Visual Servoing with Arbitrary Desired Poses

Hongxiang Yu[1], Anzhe Chen[1], Kechun Xu[1], Zhongxiang Zhou[1], Wei Jing[2], Yue Wang[1], and Rong Xiong[1]

***Abstract*—Recently, several works achieve end-to-end visual servoing (VS) for robotic manipulation by replacing traditional controller with differentiable neural networks, but lose the ability to servo arbitrary desired poses. This letter proposes a differentiable architecture for arbitrary pose servoing: a hyper-network based neural controller (HPN-NC). To achieve this, HPN-NC consists of a hyper net and a low-level controller, where the hyper net learns to generate the parameters of the low-level controller and the controller uses the 2D keypoints error for control like traditional image-based visual servoing (IBVS). HPN-NC can complete 6 degree of freedom visual servoing with large initial offset. Taking advantage of the fully differentiable nature of HPN-NC, we provide a three-stage training procedure to servo real world objects. With self-supervised end-to-end training, the performance of the integrated model can be further improved in unseen scenes and the amount of manual annotations can be significantly reduced.**

***Index Terms*—Deep Learning in Grasping and Manipulation, Transfer Learning, Visual Servoing**

I. INTRODUCTION

VISUAL servoing (VS) is a technique that uses vision feedback to guide the robot to achieve high-precision positioning. In classical VS[1], [2], [3], [4], a set of handcrafted visual features such as points, lines, contours, moments are extracted and compared with features of a pre-defined desired pose. A manually designed controller then moves the camera to the pre-defined desired pose by reducing the feature error between the desired and current poses.

With the development of deep learning, some learning-based methods have emerged to reduce the excessive manual effort in VS. [5], [6], [7], [8] use Convolutional Neural Networks (CNNs) to process the images observed at the current and the desired pose separately and estimate the relative pose, following by a position-based visual servo (PBVS) controller[1]. They get rid of expensive manual feature annotations by considering the whole image as a feature. However, the pose estimation performance depends on the similarity of the input image pairs, so the offset between the initial pose and the desired pose cannot be large [9]. [9], [10], [11] use deep learning to improve the reliability of 2D correspondences extraction followed by an image-based visual servo (IBVS) controller[1] or IBVS-based MPC[11]. The consistency of features enables VS with large pose offset. But IBVS has inherent drawbacks such as small convergence region and local minima[12], [13].

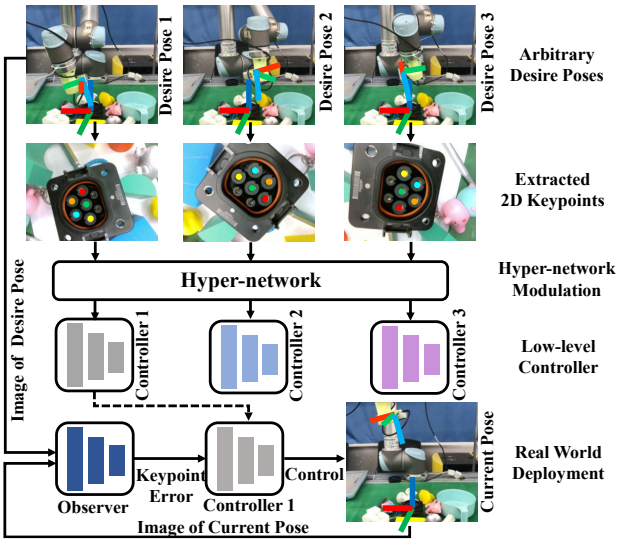

Fig. 1: The upper coordinate represents the camera's frame and the lower represents the target object's frame. Given the keypoints of desired poses, HPN generates an unique neural controller for each pose. For example, given the keypoints observed at desired Pose 1, HPN generates Controller 1 that helps the camera move to Pose 1 from arbitrary initial poses.

Recently, several works achieve self-supervised end-to-end VS for robotic manipulation tasks[14], [15], [16]. Given a real world target object, they take the 2D keypoints extracted in an unsupervised manner as features, which avoids either manual keypoint annotations and pose estimation. By replacing IBVS with a neural controller, they make the whole architecture differentiable which enables self-supervised end-to-end training. However, these methods have an obvious weakness in that they are designed to servo a fixed desired pose , thus losing the functionality of the traditional VS controller to handle arbitrary desired poses. By only taking the image observed at the current pose as input, they lack the information about the desired pose. This means that if the number of desired poses is limited, they will have to train several neural controllers and select the appropriate one according to the given desired pose. But considering a scenario that needs to change the desired pose constantly, these methods will

Manuscript received: April 18, 2023; Accepted June 6, 2023.

This paper was recommended for publication by Editor Pascal Vasseur upon evaluation of the Associate Editor and Reviewers' comments. This work was supported in part by the National Key R&D Program of China under Grant 2021ZD0114500 and the National Nature Science Foundation of China under Grant 62173293.

[1]Hongxiang Yu, Anzhe Chen, Kechun Xu, Zhongxiang Zhou, Yue Wang and Rong Xiong are with the State Key Laboratory of Industrial Control Technology and Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, Zhejiang, China. Yue Wang is the corresponding author wangyue@iipc.zju.edu.cn

[2]Wei Jing is with Alibaba Group, Hangzhou, Zhejiang, China.

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

have to train infinite controllers or to frequently fine-tune the controller to ensure VS performance. *How to implement a lightweight differentiable neural controller that can servo arbitrary desired poses remains a challenging problem.*

In this letter, we try to investigate an appropriate neural controller architecture capable of servoing arbitrary desired poses. To servo a random sampled desired poses in the 6 degree of freedom (DOF) space, it is not practical to train an infinite number of controllers. Simply adding an input that encoding information of the desired pose will inevitably enlarge network volume and prolong the inference time. In contrast, we define servoing a single desired pose as a task, and state servoing arbitrary desired poses as a multi-task learning problem. We proposing a hyper-network (HPN)[17] based neural controller (HPN-NC) to tackle this problem. HPN-NC consists of a hyper net and a low-level neural controller. Following [14], [15], [16], we use 2D keypoints as features. As shown in Fig. 1, given 2D keypoints extracted at a desired pose, the hyper net generates the parameters for the low-level controller corresponding to this desired pose. The modulated low-level controller takes the error between current and desired 2D keypoints for control inference. In this way, we can generate an unique controller for each desired pose that avoids endless fine-tuing. HPN-NC outperforms traditional IBVS and other neural controllers(NCs). To servo real world objects, we connect it with a supervised neural observer(NO) and provide a three-stage training procedure: training HPN-NC in simulation with synthetic data, training NO with manual keypoint annotations and end-to-end training the integrated model(IM:NO with NC) with all synthetic, annotated and robot's self-supervision data. Taking advantage of the fully differentiable nature of HPN-NC, IM can be further improved fully automatically in an end-to-end manner transfering to unseen scenes. Note that the amount of manual annotations can also be significantly reduced through end-to-end training, as they are only used as regularizer. Overall, the contributions of this paper are three-fold:

- We are the first work to state VS arbitrary desired poses with neural network as a multi-task learning problem. To solve this problem, we propose the HPN-NC. It outperforms other network structures when VS arbitrary desired poses both in simulation and real world experiments.
- Neural controllers can be further fine-tuned fully automatically with self-supervised end-to-end training. HPN-NC's fine-tuning ability outperforms other NCs given error-free 2D keypoints in simulation. Given imprecise 2D keypoints in unseen scene in real world, the performance of IM consists of NO and HPN-NC can be further improved.
- We also consider a situation with insufficient manual annotations. Self-supervised end-to-end training enables IM to achieve 92% success rate VS with only 30 annotations.

II. RELATED WORK

Traditional VS Methods: Classical VS methods can help the robot achieve high-precision positioning through vision feedback but highly rely on handcrafted visual features, manually labeling or recognizable QR code [3], [4] . Traditional VS controllers include IBVS[1], [18], PBVS[19], [20] and hybrid approaches [21], [22]. PBVS uses the relative pose between current and desired pose as visual feature and plans a globally asymptotically stable straight trajectory in 3D Cartesian space. IBVS uses matched keypoints on 2D image plane, which is insensitive to calibration error, but suffers from small convergence region due to the high non-linearity [12], [13]. It may meet feature loss problem[23] dealing with large initial pose offset.

Deep learning Based Methods: Deep Neural Networks[24], [25] have shown remarkable feature extraction ability on various tasks such as detection, segmentation or tracking, and also alleviate the dependency of manual effort for VS task.

Pose estimation methods[6], [5], [7] usually bypass the 2D keypoints prediction and estimate the relative pose or directly to predict control commands from image pairs observed at the current and the desired poses. [6] implements a deep nerual network to estimate relative pose between current camera pose and the desired camera pose, then performs PBVS based on the relative pose. [5] trains a convolutional neural network over the whole image with synchronised camera poses to guide the quadrotor. [7] proposes a new neural network based on a Siamese architecture which outputs the relative pose between any pair of images and realize VGA connector insertion with submillimeter accuracy. As the input image pairs have to be similar to ensure the performance of learning based pose estimator, camera pose offsets between the desired and the initial poses are limited.

Keypoint based methods[9], [10], [11], [16], [14], [15] extract 2D keypoints for the subsequent controller. [9], [10], [11] use neural networks to predict matched visual features or optical flow, then calculate control command through IBVS controller. However, IBVS has internal deficiency and may fail to servo the desired pose with large initial pose offset. Other methods[16], [14], [15] use neural controllers instead of IBVS controller. [16] learn policies that map raw image observations directly to torques at the robot's motors through deep convolutional neural networks with self-supervised learning. [14] learns the 2D keypoint representation from the image with an auto-encoder and learns the motion based on extracted keypoints. The controllers in the above two methods are trained end-to-end by self-supervised learning. The extracted keypoints can also be used to learn robot motion with end-to-end reinforcement learning [15].

III. METHODS

HPN-NC generates a unique neural controller for each desired pose in 6 DOF space. In this chapter, we first introduce implementation details of HPN-NC in Section III-A. In Section III-B, we introduce a three stage training procedure that enables HPN-NC to servo real world objects and to adapt to unseen scenes.

A. Hyper-network Neural Controller

As shown in right part of Fig. 2, HPN-NC has a hyper network (HPN, in pink) specialized in encoding the information of desired poses, and a low-level neural controller (NC,

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

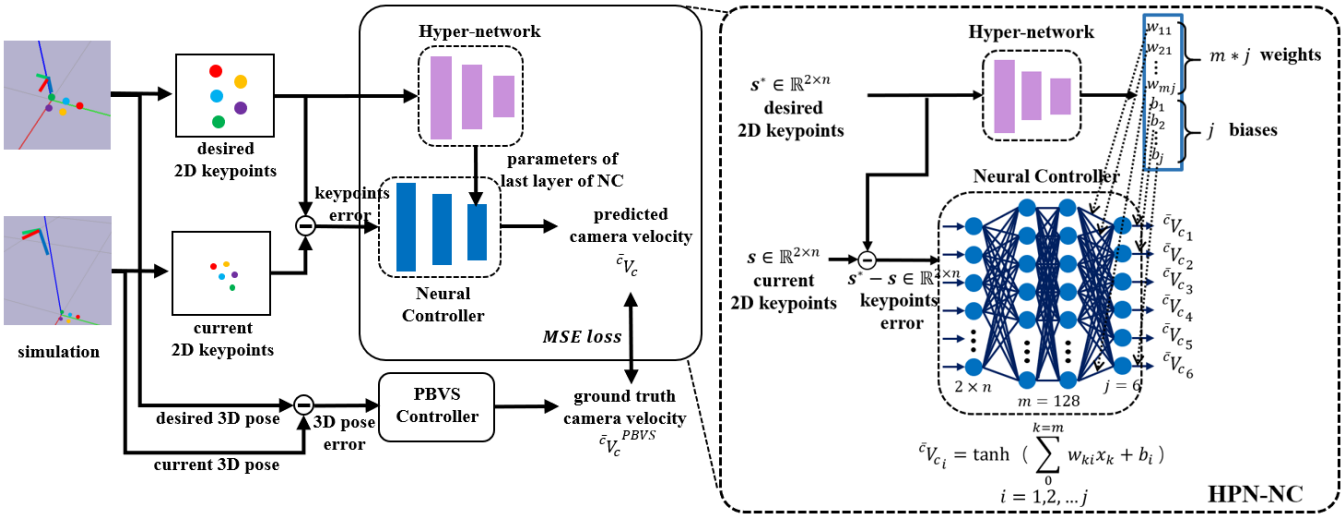

Fig. 2: The left part gives the pipeline of training HPN-NC in simulation. HPN-NC is supervised by PBVS to ensure a satisfactory servo performance. DAgger helps the model learn more quickly. The right part shows the detail of how HPN generates a NC for a given desired pose. To switch between different desired poses, HPN inferences the weights and biases of the low-level NC's last layer with the 2D keypoints obtained at the desired pose.

in blue) responsible for servoing the given pose. Both of the upper HPN and the low-level NC are lightweighted three-layer fully connected nerual networks. HPN takes the pixel coordinates $\boldsymbol{s}^* \in \mathbb{R}^{2\times n}$ of n 2D keypoints extracted at the desired pose as input and outputs the weights and biases of the last layer of the low-level NC. The last two layers of NC have 128 and 6 units respectively, so HPN outputs 128x6 weights and 6 biases. The low-level NC takes $\boldsymbol{e} = \boldsymbol{s}^* - \boldsymbol{s}$ as input just like traditional IBVS controller, which is the coordinate error of the 2D keypoints between the desired pose and the current pose. The output of NC is the 6-dimensional control command ${}^{\bar{c}}\boldsymbol{V}_c = \begin{bmatrix} v_c & \omega_c \end{bmatrix}^T \in \mathbb{R}^6$ consists of instantaneous camera linear velocity and angular velocity under camera frame. Since the last layer parameters of NC are determined by the desired pose, we are able to generate an independent controller for each desired pose and avoid any fine-tuning when switching the desired pose.

HPN-NC is trained in Pybullet simulation automatically. As shown in left part of Fig. 2, like traditional VS, we first set the virtual camera to a random desired pose under the object frame ${}^o\boldsymbol{T}_{c^*}$ to get desired 2D keypoints $\boldsymbol{s}^*$,

$$ {}^o\boldsymbol{T}_{c^*} = \begin{bmatrix} {}^o\boldsymbol{R}_{c^*} & {}^o\boldsymbol{T}_{c^*} \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \tag{1} $$

where o represents the object frame and c^* represents desired camera frame. Given the 3D model of the object and the camera intrinsic matrix, we can obtain $\boldsymbol{s}^*$ by projection. Taking the 2D keypoint $\boldsymbol{s}^*$ of the desired pose as input, HPN infers the parameters of NC's last layer, while the other two layers of NC are identical for any desired pose,

$$ \boldsymbol{\theta}_{NC} = f^{HPN}_{\boldsymbol{\theta}_{HPN}}(\boldsymbol{s}^*) \tag{2} $$

Then, we set the camera to a random initial pose ${}^o\boldsymbol{T}_c$, and get current keypoints $\boldsymbol{s}$. The low-level NC takes the keypoint error $\boldsymbol{e} = \boldsymbol{s}^* - \boldsymbol{s}$ as input and outputs the camera velocity,

$$ {}^{\bar{c}}\boldsymbol{V}_c = f^{NC}_{\boldsymbol{\theta}_{NC}}(\boldsymbol{s}^* - \boldsymbol{s}) = f^{NC}_{\boldsymbol{\theta}_{HPN},\boldsymbol{s}^*}(\boldsymbol{s}^* - \boldsymbol{s}) \tag{3} $$

PBVS provides the supervision as its trajectory in 3D space is an efficient and secure straight line. We get the supervision through desired pose ${}^o\boldsymbol{T}_{c^*}$ and current pose ${}^o\boldsymbol{T}_c$:

$$ {}^{\bar{c}}\boldsymbol{V}_c^{PBVS} = -\lambda \begin{bmatrix} {}^{c^*}\boldsymbol{R}_c^T \cdot {}^{c^*}\boldsymbol{t}_c \\ \theta u \end{bmatrix} \tag{4} $$

where θu is the axial angle of the rotation between current and the desired pose, λ is the coefficient that uniformly set as 0.4 in this work. We denote the input and output tuples to be q_{NC} and the collected training dataset to be D_{NC}:

$$ \begin{aligned} q_{NC} &\triangleq (\boldsymbol{s}^*, \boldsymbol{s}, {}^{\bar{c}}\boldsymbol{V}_c^{PBVS}) \\ D_{NC} &= \{q_{NC}\} \end{aligned} \tag{5} $$

We use MSE loss $\mathcal{L}_{NC}$ for training and dataset aggregation (DAgger) technique[26] for training acceleration:

$$ \mathcal{L}_{NC} = \| {}^{\bar{c}}\boldsymbol{V}_c^{PBVS} - f^{NC}_{\boldsymbol{\theta}_{HPN},\boldsymbol{s}^*}(\boldsymbol{s}^* - \boldsymbol{s}) \|_2^2 \tag{6} $$

The upper HPN could be a large and powerful network that has strong encoding ability. The lower NC is a lightweight fully connected network without any complicated structure. For a given desired pose, the parameter inference of NC is done before visual servoing. so the control command inference by NC during VS is efficient. Therefore, HPN-NC takes both the strong modulation for the variation of desired poses and the efficient control inference into account.

B. Real world VS with HPN-NC

We trained a neural observer (NO) to obtain 2D keypoints of ordinary objects in real world (see Fig. 4), so that it can be used as the front end of VS controllers. In order to servo objects in a large range in Cartesian space, NO needs to ensure the consistency between 2D keypoints extracted at different poses and the ground truth keypoints projected by the pre-defined 3D model on the camera plane, and also be able to adapt to various backgrounds. We train NO with manual annotated 2D keypoints. To improve its robustness,

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

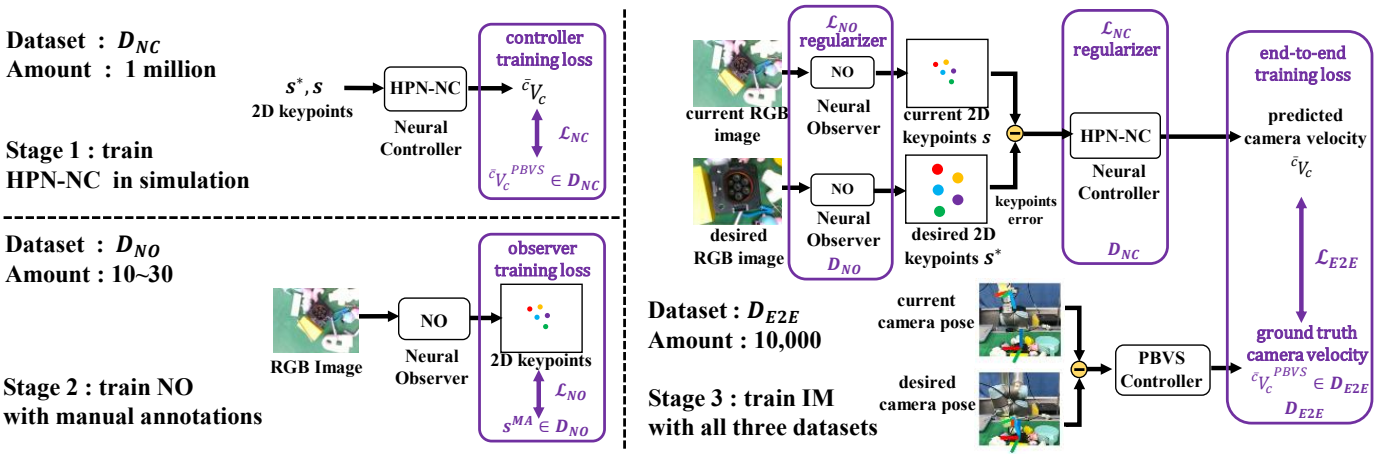

Fig. 3: The training procedure for real world VS. Image and control tuples are automatically collected to optimize the integrated model consisted of a neural observer and a neural controller. Through end-to-end training, the performance of integrated model could be improved. Note that manual annotations and the simulation data are respectively used as regularizer for observer and controller, so the amount of manual annotations can be significantly reduced.

we expand the dataset with techniques such as translation, rotation, scaling, background replacement and homography matrix stretching. But there still remains some shortcomings: due to the limited manual annotations, it is impossible to cover all the viewing angles in the workspace and NO may fail at certain camera poses, affecting the performance of VS; VS the target object in a new scenario may cause the performance degradation of NO; manual annotating 2D keypoints is costly.

These shortcomings are fatal for traditional controllers, but not for neural controllers. Being fully differentiable, the integrated model (IM, shown in Fig. 3) consists of NO and HPN-NC can be fine-tuned in an end-to-end manner in unseen scenarios. As the supervision is calculated automatically by camera poses, the training process can be self-supervised, which leads to lower data acquisition cost than manual labeling. Therefore, we can utilize a large amount of end-to-end data to improve IM, and use the manual annotations only as the regularizer. This greatly reduces the amount of the manual annotations required for training.

Stage1 Training of Controller: The training procedure of HPN-NC is described in Section III-A.

Stage2 Training of Observer: The input of the neural observer is a RGB image I, and the output is the pixel coordinates of n 2D keypoints. The backbone of NO can be SpatialConfiguration-Net(SCN)[27] or pre-trained ResNet[28]. For training dataset D_{NO}, we collect RGB images from various perspectives, distances and illuminations, and manually annotate the ground truth 2D keypoint coordinates $\boldsymbol{s}_i^{MA}$ of pre-defined 3D model. We have

$$\begin{aligned} q_{NO} &\triangleq (I, \boldsymbol{s}_i^{MA}) \ for \ i = 1, 2, ..., n \\ D_{NO} &= \{q_{NO}\} \end{aligned} \tag{7}$$

We use data augmentation techniques described above to improve the generalization of NO. NO generates a heatmap $h_i(x)$ for each keypoint i:

$$h_i(x) = f_{\boldsymbol{\theta}_{NO}}^{NO}(x) \ for \ i = 1, 2, ..., n \tag{8}$$

where x is pixel coordinates in I.

We tries to minimize the difference between the predicted heatmap and the ground truth heatmap $g_i(x)$ peaking at $\boldsymbol{s}^{MA}$. At the same time, in order to improve the accuracy of predicted keypoints, we minimize the $\boldsymbol{L_2}$ norm between the keypoint coordinates calculated by spatial-softmax operation and the ground truth coordinates $\boldsymbol{s}_i^{MA}$. Therefore,the total loss $\mathcal{L}_{NO}$ to train the observer is:

$$\begin{aligned} \mathcal{L}_{NO} = \sum_{i=1}^{n} (\gamma_h \sum_{x \in I} \| g_i(x) - h_i(x) \|_2^2 \\ + \gamma_k \left\| \boldsymbol{s}_i^{MA} - \sum_{x \in I} x h_i(x) \right\|_2^2) \end{aligned} \tag{9}$$

where $\gamma_h = 10$ and $\gamma_k = 0.00001$ are scale factors to facilitate the learning process convergence.

Stage3 End-to-end Training of Integrated Model: We fine-tune IM with large amount robot's self-supervision data in an end-to-end manner. As shown in Fig. 3, a random desired pose under the robot's base frame ${}^b\boldsymbol{T}_{c^*}$ is sampled, the robot first moves the camera to this pose. An image I^* observed at the desired poses ${}^b\boldsymbol{T}_{c^*}$ is sent to NO to obtain desired 2D keypoints $\boldsymbol{s}^*$. Afterwards the robot moves the camera to a random sampled initial pose ${}^b\boldsymbol{T}_c$ to get the initial 2D keypoints $\boldsymbol{s}$,

$$\boldsymbol{s}^* = f_{\boldsymbol{\theta}_{NO}}^{NO}(I^*), \boldsymbol{s} = f_{\boldsymbol{\theta}_{NO}}^{NO}(I) \tag{10}$$

HPN infers an unique NC for ${}^b\boldsymbol{T}_{c^*}$ according to $\boldsymbol{s}^*$,

$$\boldsymbol{\theta}_{NC} = f_{\boldsymbol{\theta}_{HPN}}^{HPN}(\boldsymbol{s}^*) \tag{11}$$

NC outputs the velocity of the camera according to the 2D keypoint error $\boldsymbol{e} = \boldsymbol{s}^* - \boldsymbol{s}$,

$${}^{\bar{c}}\boldsymbol{V}_c = f_{\boldsymbol{\theta}_{HPN}, \boldsymbol{s}^*}^{NC}(\boldsymbol{s}^* - \boldsymbol{s}) \tag{12}$$

We use DAgger to deal the out of distribution problem. When the robot is wandering in the workspace following ${}^{\bar{c}}\boldsymbol{V}_c$, it automatically collects the end-to-end (E2E) training dataset D_{E2E}. D_{E2E} is consist of image and control tuple q_{E2E} at different poses:

$$\begin{aligned} q_{E2E} &\triangleq (I, I^*, {}^{\bar{c}}\boldsymbol{V}_c^{PBVS}) \\ D_{E2E} &= \{q_{E2E}\} \end{aligned} \tag{13}$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

where ${}^{\bar{c}}\boldsymbol{V}_c^{PBVS}$ is calculated by PBVS controller according to desired pose ${}^b\boldsymbol{T}_{c^*}$ and the current pose ${}^b\boldsymbol{T}_c$:

$$ {}^{\bar{c}}\boldsymbol{V}_c^{PBVS} = -\lambda \begin{bmatrix} {}^{c^*}\boldsymbol{R}_c^T \cdot {}^{c^*}\boldsymbol{t}_c \\ \theta u \end{bmatrix} \tag{14} $$

We use MSE loss $\mathcal{L}_{E2E}$ for training:

$$ \mathcal{L}_{E2E} = \left\| {}^{\bar{c}}\boldsymbol{V}_c^{PBVS} - f^{NC}_{\boldsymbol{\theta}_{HPN},I^*}\left(f^{NO}_{\boldsymbol{\theta}_{NO}}(I^*) - f^{NO}_{\boldsymbol{\theta}_{NO}}(I)\right)\right\|_2^2 \tag{15} $$

However, there are actually infinite kinds of IMs that satisfy the constraints from D_{E2E}, results in a drift in the outputs of observer NO and controller HPN-NC. To prevent drift, we want NO and HPN-NC to satisfy the constraints from D_{NO} and D_{NC}. Thus, we use D_{NO} and D_{NC} to co-train NO and HPN-NC with D_{E2E}, so the data in D_{NO} and D_{NC} will regularize NO and HPN-NC. Since the data in D_{NO} only acts as the regularizer, only a small number of manual annotations is needed. The total loss function of Stage 3 is:

$$ \mathcal{L} = \mathcal{L}_{NC} + \mathcal{L}_{NO} + \mathcal{L}_{E2E} \tag{16} $$

Note that when a calibrated camera extrinsic matrix is given, the robotic arm can move the camera to a specified pose automatically, also the end-to-end supervision does not require any manual labeling, so the entire learning process can be fully automated.

IV. SYSTEM IMPLEMENTATION

A. Simulation Settings

An environment including a virtual camera and the target object's 3D model is built in Pybullet. In each data collecting or model evaluation episode, a random desired camera pose ${}^o\boldsymbol{T}_{c^*}$ is sampled in the space 15cm above the target object with 0 to 5cm disturbance in XYZ translation and a random initial camera pose ${}^o\boldsymbol{T}_c$ is sampled in the space 30cm above the target object with 0 to 10cm disturbance in XYZ translation. Both ${}^o\boldsymbol{T}_c$ and ${}^o\boldsymbol{T}_{c^*}$ ensure all keypoints within the camera's field of view (Fov). The maximum initial pose offset between the initial and desired pose is $\Delta\mathbf{r}_0 = ({}^{c^*}\mathbf{t}_c, \theta\mathbf{u})$: ${}^{c^*}\mathbf{t}_c = (15cm, 15cm, 30cm), \theta\mathbf{u} = (53.1°, 53.1°, 180°,)$. An episode is considered to be finished successfully if the total error between the current and the desired keypoints is lower than a specified threshold δ_f:

$$ \sum_{k=1}^{n} |u_k - u_k^*| + |v_k - v_k^*| \leq \delta_f \tag{17} $$

Before the keypoint error fully converges, there are several situations that trigger the early termination of the episode:

- Every episode has a maximum steps of δ_s with each step takes $0.1s$. An episode will finish if the keypoint error hasn't reached the threshold δ_f within δ_s steps.
- The camera walks out the workspace.
- Any 2D keypoint is out of the camera's Fov.

We use four criterias to analysis the servo performance: servo success rate (**SR**), servo efficiency (timesteps,**TS**), final rotation error (**RE**) and final translation error (**TE**). We calculate the transformation between the final camera pose and the desired pose ${}^{c^*}\boldsymbol{T}_c$.

Fig. 4: Target objects: Apriltag is used for evaluation of HPN-NC in simulation. The charging port and the toy horse are real world objects without obvious feature. Specific values of 3D models is in the supplementary materials.

Rotation Error (RE): The relative rotation ${}^{c^*}\boldsymbol{R}_c$ is converted into an axis-angle representation ${}^{c^*}\theta_c\,{}^{c^*}\boldsymbol{u}_c$. The rotation accuracy of camera is considered satisfactory if the deflection angle between the final pose and the desired pose is less than threshold δ_r.

Translation Error (TE): The translation accuracy of camera is considered satisfactory if the displacement between the final position and the desired position is less than δ_t.

If RE and TE are less than corresponding threshold, an episode would also be regarded as a successful trial. For threshold values, please refer to the supplementary material.

B. Real World Settings

The real world experiment is carried out on an UR5 robot. The settings of the real world experiment are the same as those of the simulation except for some thresholds. For specific parameter values of real world experiments, please refer to the supplementary material.

C. Target Objects and 3D Models

Compared with some pioneer works[14], [9] that need accurate objects' meshes for rendering and training, we only need to define n 3D feature points on the object and roughly measure their coordinates under object's frame as 3D model. Fig. 4 shows several objects we used as target objects. For specific coordinates of their 3D models, please refer to the supplementary material.

V. EXPERIMENTAL RESULTS

In this section, we carried out a series of experiments to evaluate our method. Simulation experiments are performed on a computer with 16 Intel(R) Core(TM) i9-9900K 3.60GHz and one NVIDIA GeForce RTX 2080 SUPER. Real world experiments are performed on a computer with 12 Intel(R) Core(TM) i7-8700 3.20GHz and one NVIDIA GeForce GTX 1060. The goals of the experiments are:

- to validate that the performance of proposed HPN-NC is better than the traditional IBVS controller and other neural controllers in multiple desired poses VS tasks.
- to validate that the integrated model (IM) is able to servo real world object with no obvious feature in unseen scenes.
- to validate that IM can further improve the servo performance in unseen scene, promote the keypoint extraction

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

ability and reduce manual annotation cost with self-supervised end-to-end training.

A. Introduction of Baseline

We compare the performance of HPN-NC with IBVS[1] and three neural controllers. These NCs have different structures but are all supervised by the same PBVS teacher. Fully connected neural controller (FCN-NC) is a three layers fully connected neural networks derived from [16], whose input is the 2D keypoints error of current and desired pose. DenseNet based neural controller (DenseNet-NC) is the controller used in [14]. DenseNet-NC's main structure is a DenseNet following with a fully connected layer. It takes the concatenated vector of current and desired keypoints as input. Auto-encoder based neural controller (AE-NC) draws on the structure of auto-encoder[29] to encode the information of the desired pose. Its low-level controller is similar to FCN-NC except for an additional input: a latent vector from the auto-encoder. For implementation details, please refer to the supplementary material.

B. Simulation Evaluations

Performance Comparison: All the NCs are trained for 100 epochs. Each epoch first runs 10 thousands data collection steps. Then NCs are trained for 500 batches with a batch size of 512. The evaluation is performed on the apriltag shown in Fig. 4. Tab. I shows the performance of controllers for 500 trials with different desired poses. Supervised by a perfect PBVS teacher, all of NCs have higher SR than IBVS. Among these NCs, HPN-NC has higher SR, servo accuracy (RE and TE) and shorter inference time. We believe that the advantage of HPN-NC stems from the fact that it uses a complex hyper network to model the modulate mechanism of different VS tasks, but only the lightweight low-level fully connected NC is used during the servo process, which ensures efficient inference. Fig. 5 shows the 2D, 3D and error trajectories for two visual servoing tasks with different desired and initial poses. All controllers have reached the desired pose. But only DenseNet-NC, AE-NC and HPN-NC complete within 200 steps. The terminal error of HPN-NC is smaller than that of DenseNet-NC and AE-NC because of stronger modulate ability. With such strong hyper net, HPN-NC's 2D and 3D trajectories are more similar to the ground truth PBVS's. For more cases, please refer to the supplementary material.

Fine-tuning with Self-supervised End-to-end Training: No matter how powerful the neural controllers are, they cannot guarantee that all desired poses can be successfully VS. But since these NCs are completely differentiable, for those desired poses that cannot be servoed, NCs can be fine-tuned through self-supervised end-to-end training. The fine-tuning process is similar to the training process described in Section III-A, except that the desired pose is fixed. We select 10 desired poses that all of the NCs in Tab. I failed to VS and compare the performance of fine-tuned NCs. As shown in Fig. 6, we compare the average SR and TS after 1-step and 3-step fine-tuning for 10 desired poses. Each step of fine-tuning runs 10 thousand data collection steps. Then the model is fine-tuned

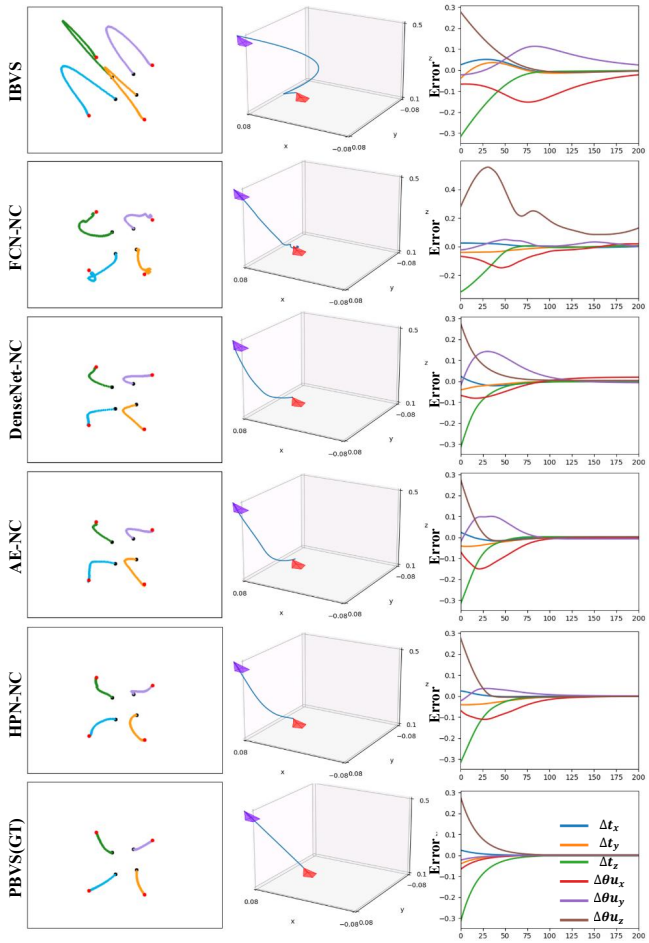

Fig. 5: 2D, 3D and error trajectories for a VS task. For 2D trajectories, black dots represent the initial keypoints and red dots represent the desired keypoints. For 3D trajectories, purple triangles represent the initial camera poses and red triangles represent the desired camera poses. Error trajectories visualize the TE and RE between the current and the desired camera poses. TE is in meters and RE is represented by the axis angle.

for 500 batches with a batch size of 512. For evaluation, the fine-tuned models try to VS the selected desired poses from 500 different initial poses. After 1-step fine-tuning, HPN-NC's average servo SR is about 90% for 10 desired poses, which is the highest among all NCs'. After 3-step fine-tuning, HPN-NC, AE-NC and FCN-NC all have high SR, but HPN-NC has less TS for about 200 steps which means it can reach the desired poses faster. In other words, only HPN-NC can achieve high success rate and high efficiency VS with efficient fine-tuning.

C. Real World Evaluations

Performance Comparison: VS real world objects in unseen scenes inevitably needs to face the recognition error of the observer. We first compare the performance of different controllers given the same neural observer. We train the NO with SpatialConfiguration-Net[27] (SCN) to extract the predefined 2D keypoints of the charging port on 1000 annotations. For specific training details, please refer to the supplementary

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

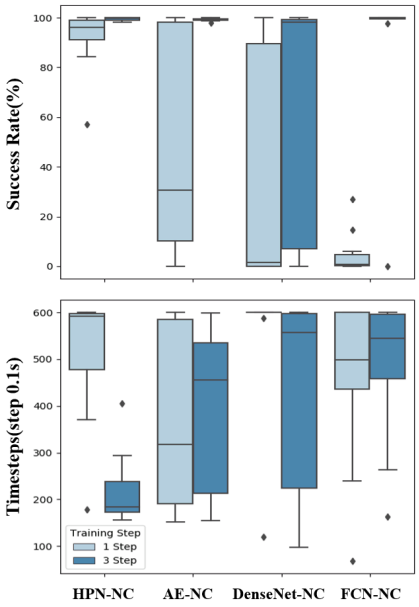

Fig. 6: We select 10 desired poses that all the NCs in Tab. I failed and compare the performance of fine-tuned NCs after 1-step and 3-step fine-tuning.

material. 2D keypoints predicted by NO are used as the input of controllers. Tab. II shows the results of different integrated models to servo 50 different desired poses. As discussed in Section III-B, limited annotations and unseen scene leads to the recognition error of NO. Although affected by recognition error, HPN-NC has the highest SR and smallest TS compared with IBVS and other NCs. The same degradation is happened to PBVS. In Row 7, the SR of PBVS which uses relative camera pose estimated by Perspective-n-Point[30] (PnP) with 2D keypoints extracted by NO, drops 10% compared with ground truth PBVS. The relative camera pose of ground truth PBVS is calculated by robot's tool center point and calibrated camera extrinsic, so it will not be affected by NO's recognition error.

Performance Improvement with Self-supervised End-to-end Training: Unfortunately, PnP is not differentiable, IM consists of NO and PBVS cannot be further promoted to deal with recognition error. Taking advantage of the fully differentiable nature of HPN-NC, IM consists of NO and HPN-NC can be improved by self-supervised end-to-end training with DAgger. IM is fine-tuned for 5 epochs. Each epoch runs 2000 data collection steps to get D_{E2E}. D_{NC}, D_{NO} and D_{E2E} are divided to be the training set with 80% data and the validation set with 20% data. Then, IM is fine-tuned for 2000 epochs with a batch size of 512,2,1 respectively for D_{NC}, D_{NO} and D_{E2E}. Lastly, a model with smallest end-to-end loss is selected for the next DAgger epoch. As shown in Tab. II, end-to-end training improves IM's SR from 78% to 98%. Servo efficiency(TS) and accuracy(RE and TE) are also improved. As discussed in Stage 3 of Section III-B, during the end-to-end training, the robot is self-supervised and no manual effort is introduced.

Manual Annotation Reduction with Self-supervised End-to-end Training: A more realistic problem is that real world objects' annotations are often insufficient due to the high production costs. Less training data introduces larger recognition error which causes VS performance degradation. Another advantage of self-supervised end-to-end training is to reduce the amount of manual annotations needed for VS. By using those annotations only as regularizer, we replace the expensive manual annotations with cheap self-supervised end-to-end data with control labels. We choose a pre-trained ResNet-18[24] as NO to avoid the failure of training NO with too little annotations. We respectively use 600, 300 and 30 pieces of annotations to train NO. As shown in Tab. III, SR of IM with PBVS(PnP) gradually decreases as the amount of manual annotations decreases. Through the end-to-end training described in Section III-B, SR of IMs can respectively be promoted to 94%(300 annotations) and 92%(30 annotations). From Fig. 7, we could find the recognition error can be reduced after end-to-end training.

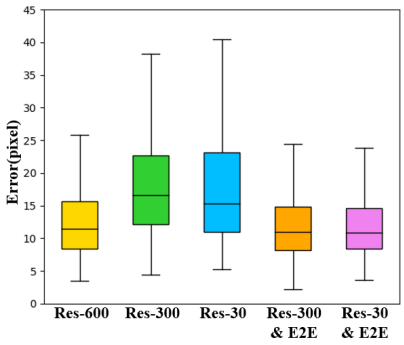

Fig. 7: The total recognition error of five 2D keypoints for the charging port. Less training data for NO introduces larger recognition error. With self-supervised end-to-end training, NO trained with 30 pieces manual annotations can reduce the recognition error to 600 piece's level.

VI. CONCLUSIONS

In this paper, we explore that hyper-network is an appropriate architecture for multiple desired poses VS. It outperforms IBVS and other neural controllers by success rate, servo efficiency, network volume, inference time and adaptation efficiency. We evaluate the proposed model in both simulation and real world experiments. For real world VS task, we propose a three-stage training procedure that can further improve the model's servo performance and reduces manual annotation amount. It's fully automatic and achieves 92% success rate with only 30 pieces of manual annotations. With the proposed training procedure, VS can be efficiently applied to similar scenarios in real world. In the future, we will address the task with more matched correspondences such as eye-to-hand VS with optical flow.

REFERENCES

[1] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *IEEE Robotics Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[2] M. Bakthavatchalam, F. Chaumette, and E. Marchand, "Photometric moments: New promising candidates for visual servoing," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 5241–5246.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

TABLE I: Performance comparison of controllers in simulation. The bold face represents the best. The underline represents the second best. To avoid failed trials to interfere with the statistic results, TS, RE and TE only count those successful trials.

Controller	SR(%)	TS(0.1s)	RE(rad)	TE(cm)	#Parameters	Inference time(ms)
IBVS[1]	17.2	**324.39±95.66**	0.039±0.036	0.861±0.835	\	\
FCN-NC[16]	27.8	510.57±156.36	0.029±0.014	0.494±0.197	**69638**	<u>0.160</u>
DenseNet-NC[14]	87	523.42±164.74	0.020±0.009	0.452±0.202	281558	1.320
AE-NC[29]	<u>89.8</u>	<u>441.59±210.74</u>	<u>0.012±0.008</u>	<u>0.327±0.192</u>	<u>71686</u>	0.164
HPN-NC(Ours)	**95**	443.9±205.33	**0.005±0.006**	**0.193±0.119**	**69638(532486)**	**0.158**

TABLE II: Real world comparison of different controllers with the same NO in unseen scene. The bold face represents the best.

Controller	SR(%)	TS(0.1s)	RE(rad)	TE(cm)
PBVS(GT)	100	60.76±26.39	0.022±0.008	0.299±0.180
IBVS[1]	30	101.20±39.70	0.221±0.194	4.942±4.631
FCN-NC[16]	34	128.31±65.73	0.155±0.126	3.263±2.745
DenseNet-NC[14]	58	120.14±62.78	0.113±0.072	2.471±1.603
AE-NC[29]	70	94.38±49.22	0.128±0.160	3.084±3.513
HPN-NC	78	93.00±47.32	0.132±0.109	2.902±2.447
PBVS(PnP)[1]	90	**60.89±23.58**	0.137±0.120	2.961±2.672
HPN-NC&E2E	**98**	90.16±56.34	**0.051±0.024**	**1.113±0.519**

TABLE III: Real world comparison of IMs trained with different amount of manual annotations. The bold face represents the best.

Integrated Model	SR(%)	TS(0.1s)	RE(rad)	TE(cm)
Res-600 PBVS(PnP)	88	90.29±34.60	0.142±0.133	2.950±2.865
Res-300 PBVS(PnP)	56	92.00±32.58	0.115±0.201	2.391±4.171
Res-30 PBVS(PnP)	30	**89.06±25.90**	0.266±0.307	4.365±3.928
Res-300 HPN-NC&E2E	**94**	119.34±53.27	0.067±0.038	1.431±0.778
Res-30 HPN-NC&E2E	92	112.48±44.05	**0.059±0.036**	**1.275±0.745**

[3] H. Shi, G. Sun, Y. Wang, and K.-S. Hwang, "Adaptive image-based visual servoing with temporary loss of the visual signal," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 1956–1965, 2018.

[4] H. Wang, B. Yang, J. Wang, X. Liang, W. Chen, and Y.-H. Liu, "Adaptive visual servoing of contour features," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 2, pp. 811–822, 2018.

[5] A. Saxena, H. Pandya, G. Kumar, A. Gaud, and K. M. Krishna, "Exploring convolutional networks for end-to-end visual servoing," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3817–3823.

[6] Q. Bateux, E. Marchand, J. Leitner, F. Chaumette, and P. Corke, "Training deep neural networks for visual servoing," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3307–3314.

[7] C. Yu, Z. Cai, H. Pham, and Q.-C. Pham, "Siamese convolutional neural network for sub-millimeter-accurate camera pose estimation and visual servoing," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 935–941.

[8] S. Felton, E. Fromont, and E. Marchand, "Siame-se (3): regression in se (3) for end-to-end visual servoing," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 14 454–14 460.

[9] N. Adrian, V.-T. Do, and Q.-C. Pham, "Dfbvs: Deep feature-based visual servo," *arXiv preprint arXiv:2201.08046*, 2022.

[10] Y. Harish, H. Pandya, A. Gaud, S. Terupally, S. Shankar, and K. M. Krishna, "Dfvs: Deep flow guided scene agnostic image based visual servoing," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9000–9006.

[11] P. Katara, Y. Harish, H. Pandya, A. Gupta, A. Sanchawala, G. Kumar, B. Bhowmick, and M. Krishna, "Deepmpcvs: Deep model predictive control for visual servoing," in *Conference on Robot Learning*. PMLR, 2021, pp. 2006–2015.

[12] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *ieee Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 313–326, 1992.

[13] R. Kelly, R. Carelli, O. Nasisi, B. Kuchen, and F. Reyes, "Stable visual servoing of camera-in-hand robotic systems," *IEEE/ASME transactions on mechatronics*, vol. 5, no. 1, pp. 39–48, 2000.

[14] E. Y. Puang, K. P. Tee, and W. Jing, "Kovis: Keypoint-based visual servoing with zero-shot sim-to-real transfer for robotics manipulation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 7527–7533.

[15] T. Wang, E. Y. Puang, M. Lee, Y. Wu, and W. Jing, "End-to-end reinforcement learning of robotic manipulation with robust keypoints representation," *arXiv preprint arXiv:2202.06027*, 2022.

[16] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[17] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," *arXiv preprint arXiv:1609.09106*, 2016.

[18] G. Allibert, E. Courtial, and F. Chaumette, "Predictive control for constrained image-based visual servoing," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 933–939, 2010.

[19] B. Thuilot, P. Martinet, L. Cordesses, and J. Gallice, "Position based visual servoing: keeping the object in the field of vision," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 2, 2002, pp. 1624–1629 vol.2.

[20] D.-H. Park, J.-H. Kwon, and I.-J. Ha, "Novel position-based visual servoing approach to robust global stability under field-of-view constraint," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 12, pp. 4735–4752, 2012.

[21] N. R. Gans and S. A. Hutchinson, "Stable visual servoing through hybrid switched-system control," *IEEE Transactions on Robotics*, vol. 23, no. 3, pp. 530–540, 2007.

[22] A. H. A. Hafez and C. Jawahar, "Visual servoing by optimization of a 2d/3d hybrid objective function," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 1691–1696.

[23] Z. Jin, J. Wu, A. Liu, W.-A. Zhang, and L. Yu, "Policy-based deep reinforcement learning for visual servoing control of mobile robots with visibility constraints," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 2, pp. 1898–1908, 2021.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[25] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[26] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[27] C. Payer, D. Štern, H. Bischof, and M. Urschler, "Integrating spatial configuration into heatmap regression based cnns for landmark localization," *Medical image analysis*, vol. 54, pp. 207–219, 2019.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[29] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow, "Interpretable transformations with encoder-decoder networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5726–5735.

[30] S. Li, C. Xu, and M. Xie, "A robust o (n) solution to the perspective-n-point problem," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1444–1450, 2012.