2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

CNS: Correspondence Encoded Neural Image Servo Policy

Anzhe Chen†, Hongxiang Yu†, Yue Wang*, Rong Xiong

***Abstract*— Image servo is an indispensable technique in robotic applications that helps to achieve high precision positioning. The intermediate representation of image servo policy is important to sensor input abstraction and policy output guidance. Classical approaches achieve high precision but require clean keypoint correspondence, and suffer from limited convergence basin or weak feature error robustness. Recent learning-based methods achieve moderate precision and large convergence basin on specific scenes but face issues when generalizing to novel environments. In this paper, we encode keypoints and correspondence into a graph and use graph neural network as architecture of controller. This design utilizes both advantages: generalizable intermediate representation from keypoint correspondence and strong modeling ability from neural network. Other techniques including realistic data generation, feature clustering and distance decoupling are proposed to further improve efficiency, precision and generalization. Experiments in simulation and real-world verify the effectiveness of our method in speed (maximum 40fps along with observer), precision (<0.3° and sub-millimeter accuracy) and generalization (sim-to-real without fine-tuning). Project homepage (full paper with supplementary text, video and code): https://hhcaz.github.io/CNS-home.**

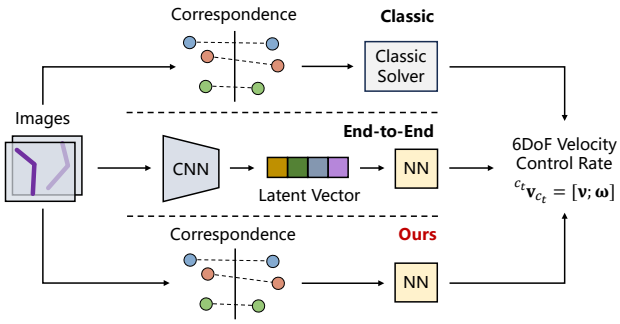
I. Introduction

Image servo is an indispensable technique for robotic applications such as navigation and manipulation [1], [2], [3], [4]. Given the current image, the robot needs to adjust its pose to make it consistent with the desired image to achieve high precision positioning.

Traditional methods extract keypoint correspondence between current and desired images, and derive velocity control directly from keypoints error (image-based visual servoing, IBVS) or estimate the relative pose transformation with extra object model (position-based visual servoing, PBVS). These methods achieve high servo precision, but suffer from either small convergence basin and error correspondence (IBVS), or imprecise object model and camera intrinsic (PBVS) [5], [6]. With the development of deep learning, it has become a trend to empower image servo with data.

Image servo is a sensory-act process, its intermediate representation is important in sensor input abstraction and policy output guidance. From this perspective, IBVS adopts keypoint correspondence as explicit representation. Following this line, prior works [6], [7], [8], [9] try to improve the accuracy and density of correspondence, or fine-tune the parameter of IBVS to improve the overall performance of IBVS based methods. However, these methods cannot overcome the intrinsic problem of IBVS.

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 (Grant 62173293 and 62373322) and the Innovation and Development Special Fund of the Hangzhou Chengxi Sci-tech Innovation Corridor. Anzhe Chen, Hongxiang Yu, Yue Wang and Rong Xiong are with Zhejiang University, Hangzhou, Zhejiang, China. Anzhe Chen, Hongxiang Yu contributed equally to this work. Yue Wang is the corresponding author wangyue@iipc.zju.edu.cn



	Generalization	Precision	Convergence
Classic	Good	High	Weak
End-to-End	Weak	Moderate	Good
Ours	Good	High	Good

Fig. 1: We utilize explicit correspondence and neural policy, endowing image servo with generalization, high precision and large convergence basin.

Another line of methods [10], [11], [12], [13] investigate the implicit representations. They encode image to latent vectors which naturally avoid explicit error correspondence and predict velocity in an end-to-end manner supervised by PBVS, which has a larger convergence basin. These methods converge well and achieve comparable precision with IBVS in specific scenes seen in training, however, cannot generalize well to novel scenes because some spurious scene-specific features are learned.

In this paper, we introduce Correspondence encoded Neural image Servo policy (CNS), which utilizes explicit correspondence with neural policy to combine the both advantages (Fig.1). We design the architecture, randomization and introduce several techniques to overcome the problems of explicit correspondence and neural policy to achieve:

i) High precision: We model arbitrary number of keypoints and the intermittent correspondence as a graph with time-variant structure and intuitively, CNS is built on a graph neural network (GNN). We use clustering and attentional aggregation to resolve error correspondence. We also simulate the error correspondence when randomization in training as data augmentation to further improve the error tolerance;

ii) Large convergence basin: Our neural policy is supervised by PBVS which intrinsically has larger convergence basin than IBVS. Moreover, we introduce graph convolutional gated recurrent unit to implicitly model scene structure which further improves the convergence and robustness to intermittent correspondence;

iii) Generalization: As keypoint correspondence isolates the appearance of the image from the neural policy, our

model naturally generalizes to novel scenes. Besides, we predict a distance decoupled velocity which prevents the neural policy overfitting to scenes of specific scale that used in training. Several randomization techniques are also introduced for further improving the generalization.

We verify the effectiveness of CNS both in simulation and real-world. The policy trained in simulation can be directly transferred to novel scenes in real-world without any fine-tuning, relieving the burden of deployment.

II. Related Works

Classical Visual Servo: Traditional visual servo includes IBVS, PBVS and hybrid approaches. IBVS [14], [15] use geometric features such as keypoint correspondence extracted from images which is robust to calibration and model errors. However, it has limited convergence basin (feature loss problem [6], unpredictable 3D trajectory [5], Jacobian singularities or local minima [16]). PBVS uses camera's 3D poses as features and is globally asymptotically stable. However, it requires precise camera intrinsic and 3D models of observed objects for pose estimation. Hybrid approaches switch between [17] or combine [5], [18] the two methods to utilize the both advantages. Overall, these methods are general and usually highly precise with clean correspondence, but limited either in convergence basin or robustness to feature error.

Learning Based Visual Servo: To improve model's robustness and convergence, learning based methods are introduced, they can be classified into three categories.

The first category tries to improve the observer, that is, the quality (correctness, density) of keypoint correspondence [7] with modern learning based feature matching methods [19], [20], [21], or optical-flow estimation methods [8], [9], and still use classic controller for servoing. These methods can generalize to novel scenes but do not overcome the intrinsic problem of controllers.

The second category focuses on improving the controller or the both. Given 3D mesh of objects, [3] trains the neural observer on specific scenes to provide accurate keypoints for the neural controller to achieve high precision servoing. However, the trained observer and controller work only for seen scenes and fixed desired pose. [22] treats the arbitrary desired pose servoing as a multi-task learning problem and use a hyper-net to generate weights for neural controller according to the given desired pose. However, it is still not general enough since the number of keypoints is fixed and it can not handle temporarily missing keypoints.

The last category doesn't strictly distinguish between observers and controllers but predicts velocity or pose in an end-to-end manner. They [10], [11], [12], [13] encode the current and desired images into latent vectors and predict the velocity or pose with a MLP. These methods could achieve comparable precision with classic methods and is robust to image occlusions or other feature error, however, cannot generalize to novel scenes since they adopt a pure convolution structure which is not rotation-equivariant.

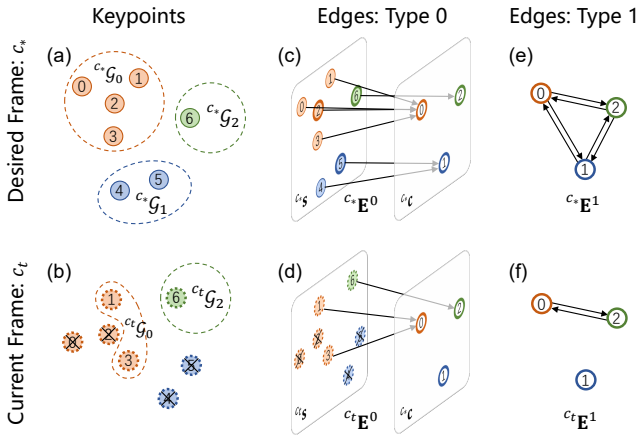


Fig. 2: A simple case illustration of the graph representation of 7 keypoints and correspondence. Keypoints are clustered into groups according to positions. Edges are categorized into 2 types for intra-cluster aggregation and inter-cluster information mutation.

Our CNS takes matched keypoints provided by any detector-based feature matching methods (*e.g.*, SIFT, ORB, AKAZE, SuperGlue [20]) and uses GNNs to model arbitrary number of keypoints and intermittent correspondence. CNS benefits from existing generalizable observers with a compatible network architecture as well as error tolerant modeling ability from deep learning.

Graph Neural Network: Graph neural network is popular in point cloud processing [23], [24], [25], [26], [27], [28] for its permutation-invariant nature to handle irregular data. Yet various mechanisms have been developed to improve the performance of GNNs. We briefly introduce relevant structures used in our work. [25] proposes EdgeConv to perform graph convolution on kNN graphs. [28] incorporates the self-attention to attentionally aggregates neighbor node embeddings. [29] introduces GraphNorm to accelerate training GNNs.

III. Architecture

A. Keypoints and Correspondence as Graph

Enrolling GNN for visual servo starts with the graph representation of keypoints and their correspondence. We assign grpah nodes' positions and initial embedding with the keypoint positions in normalized image plane and categorize edges into two types: one for intra-cluster embedding aggregation and the other for inter-cluster information mutation.

Notation: To formally define the graph, we first define notations. The left superscript c_* means variables obtained from desired pose, whereas c_t from current pose. $\mathbf{s}$ represents keypoint positions in the normalized image plane. ${}^{c_*}\mathbf{s}_i$ represents the i-th keypoint of total N keypoints in desired pose. $\mathcal{G}$ is a group containing keypoint indices belonging to a specific cluster (we use Affinity-Propagation algorithm for clustering right after the extraction of ${}^{c_*}\mathbf{s}$). ${}^{c_*}\mathcal{G}_i$ contains indices of i-th cluster (total N_c clusters) in desired pose. $\mathbf{c}$ represents the center keypoint of a cluster. Cluster center keypoint $\mathbf{c}_i$ for i-th cluster is chosen as the point closest to the mean keypoints position of that cluster: $\mathbf{c}_i = \mathrm{argmin}_{\mathbf{s}_j} \|\mathbf{s}_j - \frac{1}{N_{c_i}} \sum_{k \in \mathcal{G}_i} \mathbf{s}_k\|$, where N_{c_i} is the number of contained keypoints of i-th

cluster. $\mathbf{E}$ represents edges where $\mathbf{E}[i][j] = 1$ means a directed connection from j-th source node to i-th target node. Node's position is chosen from $\mathbf{s}$ or $\mathbf{c}$.

Case Illustration: We present the graph structure by referring to a case shown in Fig.2. Suppose we have detected total 7 keypoints (circle with solid edge filled with a number indicating its index) in desired pose (Fig.2a). The keypoints are grouped into 3 clusters: ${}^{c_*}\mathcal{G}_0 = \{0, 1, 2, 3\}$ (red circles), ${}^{c_*}\mathcal{G}_1 = \{4, 5\}$ (blue circles), and ${}^{c_*}\mathcal{G}_2 = \{6\}$ (the green circle). For ${}^{c_*}\mathcal{G}_0$, ${}^{c_*}\mathbf{s}_2$ is chosen as the center point ${}^{c_*}\mathbf{c}_0$. For ${}^{c_*}\mathcal{G}_1$, since ${}^{c_*}\mathbf{s}_4$ and ${}^{c_*}\mathbf{s}_5$ have equal distances to the mean position, we just random pick one as center point, and here we choose ${}^{c_*}\mathbf{s}_5$. For ${}^{c_*}\mathcal{G}_2$, it contains only one keypoint, therefore the center is chosen as that keypoint ${}^{c_*}\mathbf{s}_6$. Edges ${}^{c_*}\mathbf{E}^0$ define connections from each keypoint to center keypoint of its belonging cluster (Fig.2c), edges ${}^{c_*}\mathbf{E}^1$ define connections among each cluster center keypoints (Fig.2e):

$$ {}^{c_*}\mathbf{E}^0 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad {}^{c_*}\mathbf{E}^1 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \tag{1} $$

where ${}^{c_*}\mathbf{E}^0$ connect ${}^{c_*}\mathbf{s}$ (source) to ${}^{c_*}\mathbf{c}$ (target) (Fig.2c), while for ${}^{c_*}\mathbf{E}^1$, source and target nodes are both ${}^{c_*}\mathbf{c}$ (Fig.2e).

In current pose, suppose the corresponding keypoints of ${}^{c_*}\mathbf{s}_0$, ${}^{c_*}\mathbf{s}_2$, ${}^{c_*}\mathbf{s}_4$ and ${}^{c_*}\mathbf{s}_5$ are missing (Fig.2b). As a result, positions and node embeddings of ${}^{c_t}\mathbf{s}_0$, ${}^{c_t}\mathbf{s}_2$, ${}^{c_t}\mathbf{s}_4$ and ${}^{c_t}\mathbf{s}_5$ are meaningless, they shouldn't participate in the intra-cluster embedding aggregation. Therefore, we drop them from the index groups $\mathcal{G}$ and edges $\mathbf{E}^0$, yielding ${}^{c_t}\mathcal{G}_0 = \{1, 3\}$, ${}^{c_t}\mathcal{G}_1 = \varnothing$, ${}^{c_t}\mathcal{G}_2 = \{6\}$. Since none of keypoints in $\mathcal{G}_1$ is observed, the cluster embedding on center keypoint is meaningless and shouldn't participate in inter-cluster information mutation, we need to drop them from edges $\mathbf{E}^1$, yielding:

$$ {}^{c_t}\mathbf{E}^0 = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad {}^{c_t}\mathbf{E}^1 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{2} $$

Note that we still use ${}^{c_*}\mathbf{c}$ as target nodes for edges ${}^{c_t}\mathbf{E}^0$ (Fig. 2d), because the cluster center keypoint at current pose may not be observed either. The source nodes and target nodes are also ${}^{c_*}\mathbf{c}$ for edges ${}^{c_t}\mathbf{E}^1$ (Fig.2f). Fig.7 shows the evolution of graph structure in real-world experiments.

Clustering for Efficiency: If no clustering is applied, each keypoint becomes an individual cluster, we then have ${}^{c_*}\mathbf{E}^0 = \mathbf{I}_{N\times N}$ and ${}^{c_*}\mathbf{E}^1 = 1 - \mathbf{I}_{N\times N}$. In such condition, any graph convolution takes ${}^{c_*}\mathbf{E}^0$ is simply a MLP without aggregation, and ${}^{c_*}\mathbf{E}^1$ defines dense connected edges which consumes much memory and time for graph convolution when numerous keypoints are detected. Thus clustering obviously saves memory and inference time.

Clustering for Higher Precision: If no clustering is applied, each keypoint contributes equally to the final control rate, so does the noisy and mismatched keypoint. While clustering are used to achieve intra-cluster embedding aggregation by an attentional graph convolution (later described in section III-B), which has the possibility to lower the contribution of these keypoints.

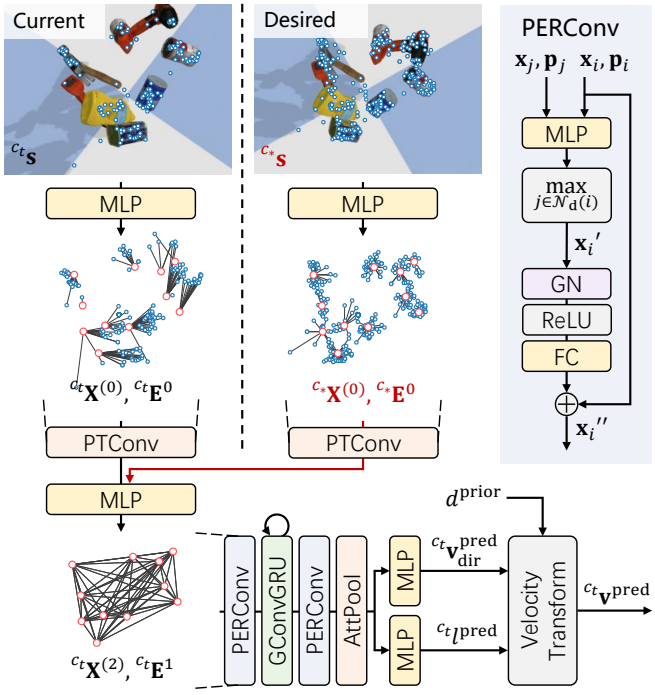


Fig. 3: Architecture of CNS. CNS takes over spatio-temporal features of keypoints and predicts distance decoupled velocity. PTConv aggregates intra-cluster embeddings with edge ${}^{c_t}\mathbf{E}^0$ or ${}^{c_*}\mathbf{E}^0$, PERConvs are used to achieve inter-cluster information mutation with edge ${}^{c_t}\mathbf{E}^1$. GConvGRU utilizes historical observations to implicitly model scene structure, brining larger convergence basin than IBVS and generating more smooth control.

B. Network Architecture

By encoding keypoints and correspondence as a graph, we naturally build the policy with GNNs (Fig.3). To achieve intra-cluster node embedding aggregation, we use Point-Transformer convolution [28] (denote as PTConv in Fig.3) to aggregate features of keypoints at desired pose and current pose. The aggregated features are subtracted, concatenated and fused by a MLP. Since low grain features have been aggregated to cluster centers, further convolutions can be conducted only on these centers which significantly reduce the computing complexity when numerous keypoints are encountered. We propose to use two point-edge-residual convolution (PERConv) layers to propagate information among cluster centers.

Graph Convolutional Gated Recurrent Unit: Keypoints provided by observers may be temporarily missing and jitter in image plane. A pure feedforward network structure may suffer from noises in a sequential decision task, therefore, it's natural to incorporate temporal modeling ability in our controller. We thus propose graph convolutional GRU (GConvGRU) built upon PERConv and GRU as Eq.7 (Supplementary Material, section VII-B). Our GConvGRU is different from gated graph convolution proposed by [30]. The latter runs graph convolution and GRU cell sequentially, while in our GConvGRU, graph convolution directly participates in the predictions of gates and the hidden state.

Network Predicition: CNS predicts distance decoupled velocity for better generalization. Specifically, it predicts the direction $\mathbf{v}_{\text{dir}}^{\text{pred}} = [\boldsymbol{\nu}_{\text{dir}}^{\text{pred}}; \boldsymbol{\omega}_{\text{dir}}^{\text{pred}}]$ and norm $\mathcal{T}(l^{\text{pred}})$ of the

distance decoupled velocity. Following the IBVS requiring depth value (can be roughly estimated) to calculate velocity, we also need a distance prior (only a scalar) to scale the linear velocity of network predictions for actual control:

$$\mathbf{v}^{\text{pred}} = \frac{\mathbf{v}_{\text{dir}}^{\text{pred}}}{\|\mathbf{v}_{\text{dir}}^{\text{pred}}\|_2}\mathcal{T}(l^{\text{pred}}) \cdot d; \quad \boldsymbol{\omega}^{\text{pred}} = \frac{\boldsymbol{\omega}_{\text{dir}}^{\text{pred}}}{\|\mathbf{v}_{\text{dir}}^{\text{pred}}\|_2}\mathcal{T}(l^{\text{pred}}) \quad (3)$$

where $\mathcal{T}(\cdot) = 1 + \text{ELU}(\cdot)$. $\mathcal{T}$ decays exponentially to zero when input is negative, encouraging the network to predict more accurate velocity when close to the desired pose to avoid damping behavior. When input is positive, $\mathcal{T}$ behaves as a linear function, preventing the network predicting exaggerated large velocity.

C. Loss

We use PBVS [14] as supervision since poses are always known in simulation. We divide the linear velocity of supervision by ground truth distance prior $d^{\text{gt}} = \| {}^{c_*}\mathbf{p}_o\|_2$ (distance from scene center to camera at desired pose) to obtain the distance decoupled representation of the supervision: $\mathbf{v}_{\text{dd}}^{\text{gt}} = [\mathbf{v}^{\text{gt}}/d^{\text{gt}}; \boldsymbol{\omega}^{\text{gt}}]$. We supervise the direction and the norm of the distance decoupled velocity separately:

$$\begin{aligned} \mathcal{L}_{\text{dir}} &= 1 - \text{CosineSimilarity}(\mathbf{v}_{\text{dir}}^{\text{pred}}, \mathbf{v}_{\text{dd}}^{\text{gt}}) \\ \mathcal{L}_{\text{norm}} &= \text{MSE}\left(l^{\text{pred}}, \mathcal{T}^{-1}\left(\|\mathbf{v}_{\text{dd}}^{\text{gt}}\|_2\right)\right) \end{aligned} \quad (4)$$

with the final servo loss as: $\mathcal{L}_{\text{servo}} = \mathcal{L}_{\text{dir}} + 0.1\mathcal{L}_{\text{norm}}$.

IV. Randomization in Training

Thanks to the graph based representation, we need to only sample keypoints rather than render images for faster training. It also enables us to directly simulate the non-idealities of keypoints and correspondence regradless of detailed image appearance for generalization.

A. Keypoints Distribution Simulation

Empirically, we assume the distribution of keypoints inversely projected to 3D space as a union of multiple bounded uniform distributions. Denoting $\mathcal{S}(n, x, y, z, h, a, b)$ as a randomization scheme to uniformly sample n points in a elliptic cylinder located at center (x, y, z) with height h, major axis a and minor axis b. Given total number of points N in scene with maximum scene boundary size r, we first determine the number of clusters to generate as N_c. Each cluster i is randomly assigned with n_i points with cluster center coordinates as $\{(x_i^c, y_i^c, z_i^c)\}_{i=1}^{N_c}$. Afterwards, points in i-th cluster are generated: $\mathbf{P}_i = \{(x_j, y_j, z_j)\}_{j=1}^{n_i} = \mathcal{S}(n_i, x_i^c, y_i^c, z_i^c, 0.1r, a, b)$. Random rotation around cluster's center is applied to all points belonging to the same cluster to mimic the various object surfaces in the scene. Finally, the residual points are uniformly distributed in the whole scene: $\mathbf{P}_0 = \mathcal{S}(N - \sum_i n_i, 0, 0, 0, 0.5r, r, r)$. The overall 3D points set is the union: $\mathbf{P} \in \mathbb{R}^{N\times 3} = \cup_{i=0}^{N_c}\mathbf{P}_i$. Given camera's extrinsic ${}_w^c\mathbf{T}$ and intrinsic $\mathbf{K}$, keypoints in normalized image plane $\mathbf{S} \in \mathbb{R}^{N\times 2}$ can be obtained by projection: $\mathbf{S} = \text{Project}(\mathbf{K}, {}_w^c\mathbf{T}, \mathbf{P})$. Fig.4 shows an example of generated points and their projection in the normalized image plane. Detailed values of hyper-parameters to generate points are listed in Supplementary Material, section VII-A.

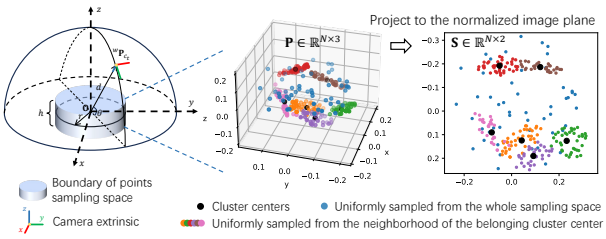


Fig. 4: Keypoints sampling and projection. The blue cylinder defines the boundary of points sampling space in training, or ranges of positions of objects from YCB datasets [31] in evaluation (in environment E_{render} described later in section V-A). After sampling points in 3D space, we project them to the normalized image plane as the detected keypoints.

B. Non-ideal Correspondence

Only a subset of keypoints from desired pose frame can find their correspondence in current pose when facing object occlusions and large perspective change. Since the perspective change is resulted from camera motion, the initial observable region may shift to other areas when moving from initial pose to desired pose. To simulate the shifting observable region and missing keypoints behavior, we assign an observable probability p_i and a jitter time constant τ_i to each keypoint $\mathbf{s}_i \in \mathbf{S}$. We define three Gaussian kernels in normalized image plane where keypoints close to kernel centers have higher p_i:

$$p_i = \max(\{\kappa(\|\mathbf{s}_i - \mathbf{k}_j\|_2, 0.3r)\}_{j=1}^3) \quad (5)$$

where $\kappa(\mu, \sigma) = \exp(-\mu^2/(2\sigma^2))$, $\mathbf{s}_i = (x_i^{\text{kp}}, y_i^{\text{kp}})$ is the coordinate of i-th keypoint and $\mathbf{k}_j = (x_j^{\text{kc}}, y_j^{\text{kc}})$ is the coordinate of j-th kernel center. The kernel center changes with camera motion to produce observable region shifting behavior. This can be implemented by sampling x_j^{kc} and y_j^{kc} from 1D Perlin noise to realize smooth and random shifting trajectory:

$$x_j^{\text{kc}} = \text{Perlin1D}\left(\frac{0.2}{\pi}\|(\theta\mathbf{u})|_{t_0}^{t_{\text{now}}}\|_2 + \frac{0.2}{r}\|\mathbf{t}|_{t_0}^{t_{\text{now}}}\|_2\right) \quad (6)$$

where $(\theta\mathbf{u})|_{t_0}^{t_{\text{now}}}$ denotes axis-angle representation of relative rotation from initial simulation time step t_0 to current t_{now}, $\mathbf{t}|_{t_0}^{t_{\text{now}}}$ denotes the relative translation from t_0 to t_{now}.

Given observable probability p_i derived from Eq.5 and $\tau_i \sim \mathcal{U}(0.5, 5)$, a keypoint reverses its state via a kinetic Monte Carlo (KMC) process with average time from observable to missing $\tau_i^{\text{o2m}} = p_i\tau_i$ and average time from missing to observable $\tau_i^{\text{m2o}} = (1 - p_i)\tau_i$.

V. Experimental Results

A. Simulation Environment Setup

We introduce two benchmark environments and stopping criterion for fairly benchmarking servo policies. The first environment named E_{render} randomly places 8 to 12 objects from YCB datasets [31] in a scene (Fig.9a in Supplementary Material). We generate total 150 scenes and randomly sample an initial pose and a desired pose for each scene. Image observed from specific pose is rendered by PyBullet simulation engine. The second environment named E_{affine} use 2D image as scene (Fig.9b in Supplementary Material), similar as [11],

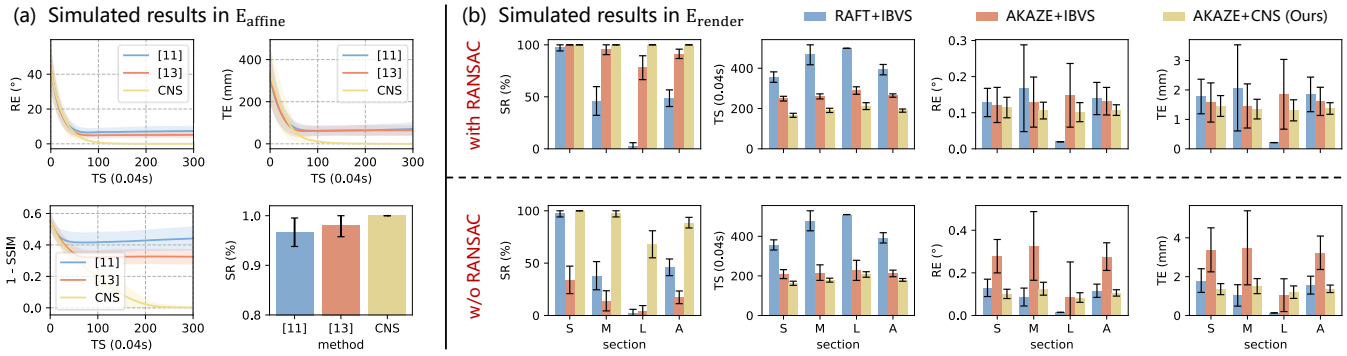


Fig. 5: Comparisons in simulation. (a) Comparison with [11] and [13] in environment E_{affine}, with average initial RE of 44.69° and TE of 284.6mm. Each statistic is calculated from 150 runs. We assume a success servo episode when the final RE is less than 10° and TE less than 10cm. (b) Comparison with IBVS based methods in environment E_{render} with or without RANSAC. Each statistic is calculated from 150 runs. "S", "M", "L" denote sections (each containing 50 scenes) of small, median and large initial RE (with average of 24.06°, 67.38° and 136.46°), respectively. "A" denotes all the scenes with average initial RE of 75.96° and TE of 247.9mm. We assume a success servo episode when the final RE is less than 3° and TE less than 3cm.

TABLE I: Comparison with IBVS in real-world. Each statistic is calculated from 60 runs with average initial RE of 86.56° and TE of 148.8mm. Both IBVS and CNS use SIFT as the observer. We assume a success servo episode when the final RE is less than 3° and TE less than 3cm.

	Scene-Easy, with RANSAC		Scene-Easy, w/o RANSAC		Scene-Hard, with RANSAC	
	IBVS	CNS (ours)	IBVS	CNS (ours)	IBVS	CNS (ours)
SR (%) (95% ci)	98.33 (95.90, 100)	**100**	8.33 (1.34, 15.33)	**75.00** (64.04. 85.96)	70.00 (58.40, 81.60)	**96.67** (92.12, 100)
TS (0.04s)	**145.3**±46.3	158.9±44.0	358.8±65.1	**136.6**±49.4	232.5±45.2	**217.8**±51.2
RE (°)	0.147±0.088	**0.053**±0.046	1.368±0.429	**0.067**±0.033	0.523±0.476	**0.101**±0.125
TE (mm)	0.843±0.505	**0.308**±0.162	8.256±2.657	**0.385**±0.156	2.839±2.867	**0.367**±0.438

[13] do. Since the scene is pure 2D, the resulting rendered image is affined from the original image. As for the stopping criterion, since keypoints are already extracted at current pose and desired pose for CNS and IBVS, we therefore use average keypoint position error as criterion and stop current servo process when error is below certain threshold and no longer decreases for 20 steps. While for [11] and [13], we use structural similarity (SSIM) [32] error between current image and desired image as criterion.

Metrics: We use the following metrics to evaluate the performance of servo policies: (1) SR (success ratio, we also add 95% confidence interval in tables), (2) TS (time steps to convergence, each time step lasts for 0.04s), (3) RE (rotation error), (4) TE (translation error), (5) mOT (mean time cost on observer in each frame), (6) mCT (mean time cost on controller in each frame), (7) mTT (mean total time cost on each frame, the summation of mOT and mCT).

B. Simulation Results

We benchmark our CNS and those implicit models in E_{affine} with the same image used by [11] and [13]. All the models achieve high success ratio while our model achieves best servo precision (Fig.5a). The servo precision of [11] and [13] is limited because they use a MLP taking features from the last convolution layer of pre-trained CNN model (specifically trained for image classification) for servo. However, difference on the most coarse feature map struggles to capture minor difference from original image scale since the feature map has been down-sampled multiple times, resulting limited servo precision.

We compare our CNS with IBVS in E_{render}. Scenes are divided into three sections (S, M, L) with increasing rotational deviation between initial and desired pose. Section A means the average result of all three sections. Results (Fig.5b) indicate that IBVS controller fails more on scenes with large initial RE. Compared with AKAZE+IBVS, RAFT[33]+IBVS performs worse when initial RE increases, we assume the reason is that data for training RAFT [33] may not contain such large rotation deviation, resulting weak generalization to these conditions. Our model success in all scenes and achieves highest servo precision in most scenes. Previous experiments use RANSAC as the additional post-processing step to reject outlier correspondences. When this geometric verification step is removed (Fig.5b, "w/o RANSAC"), AKAZE+IBVS shows significant performance drop while our model still preserves high success ratio and servo percision, which is more robust to mismatches.

C. Real-world Environment Setup

We compare our CNS with traditional high precision IBVS in two real-world scenes: Scene-Easy and Scene-Hard (Fig.9c in Supplementary Material). We uniformly scatter 20 objects (this results in rich keypoints) in Scene-Easy, and randomly sample 60 initial-desired pose pairs for statistics. In Scene-Hard, we use the same initial-desired pose pairs as those in Scene-Easy, but there are only 3 objects which are often partially observed at the initial or desired poses (Fig.6). Moreover, the red box locates much higher than the other two yellow pliers.

D. Real-world Results

Results show that CNS achieves higher success ratio and precision than IBVS in Scene-Easy (Table I). In Scene-Hard, although CNS does not achieve 100% SR, it is much higher than IBVS. We also remove the RANSAC to inspect models' dependency on post-processing. Results show CNS is more

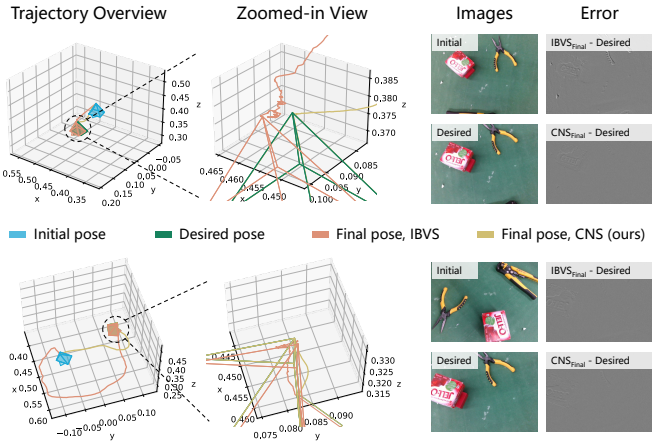


Fig. 6: Sampled success cases of IBVS and CNS in real-world experiments with Scene-Hard setup. We visualize two cases. In each case, we plot the overview trajectories and their zoomed-in views around the desired pose. Next to the 3D trajectories are images obtained from initial and desired poses, and the gray-scale images representing the photometric error between the desired image and images obtained from the final poses guided by IBVS and CNS.

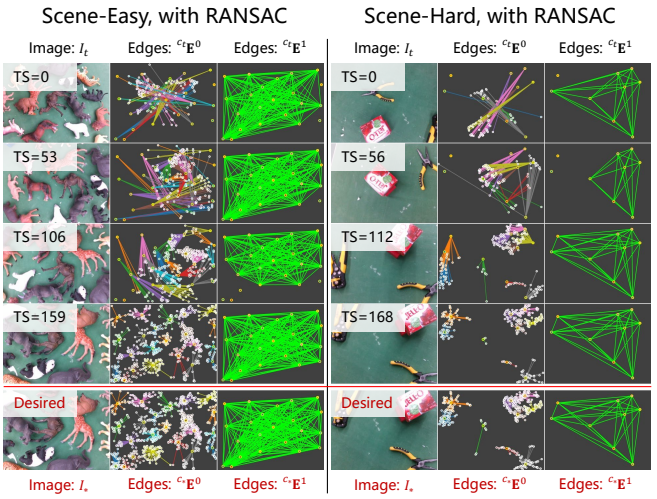


Fig. 7: Two samples of first perspective sequence of CNS in Scene-Easy (left) and Scene-Hard (right). In each scene, the first column shows images from current poses (except the last image from the desired pose), the second column shows the structure of edges ${}^{c_t}\mathbf{E}^0$ (except the last row to be ${}^{c_*}\mathbf{E}^0$) and the third column shows ${}^{c_t}\mathbf{E}^1$ (except the last row to be ${}^{c_*}\mathbf{E}^1$). In edges ${}^{c_t}\mathbf{E}^0$ or ${}^{c_*}\mathbf{E}^0$, points with white border are ${}^{c_t}\mathbf{s}$ and ${}^{c_*}\mathbf{s}$ respectively, while points with yellow border are always ${}^{c_*}\mathbf{c}$, as illustrated in section III-A.

robust to noisy and mismatched keypoints than IBVS while IBVS almost fails in all scenes. Moreover, CNS generate more smooth trajectories than IBVS (Fig.6).

E. Ablation Study on Network Structure

To evaluate the effectiveness of clustering, we set the structure illustrated in section III-B as base CNS model and replace the PTConv of base CNS with a point-wise MLP, resulting in a huge performance degradation in convergence time, servo precision and computing efficiency (Table II, column "-Cluster"). Compared with CNS without GConvGRU or using GGNN proposed by [30], the base model improves over 50% in precision and reaches 100% success rate in $\mathrm{E}_{\mathrm{render}}$, with a slight increase ($\sim$10%) on network inference time (Table II, column "-GConvGRU" and "+GGNN").

TABLE II: Ablation study on network structure. Each statistic is obtained from 150 runs in simulation environment $\mathrm{E}_{\mathrm{render}}$. Clustering boosts the convergence time, precision and computing efficiency, and our GConvGRU further improves the convergence and precision. (-Cluster: replace PTConv with a point-wise MLP; -GConvGRU: replace GConvGRU with a point-wise MLP; +GGNN: replace GConvGRU with GGNN [30].)

	CNS (base)	-Cluster	-GConvGRU	+GGNN[30]
SR (%)	**100**	99.33	99.33	97.33
TS (0.04s)	207.8±43.8	378.7±183	218.6±62.0	**203.4**±118
RE (°)	**0.12**±0.11	0.72±0.62	0.23±0.29	0.24±0.28
TE (mm)	**1.39**±1.40	9.29±9.18	2.92±3.60	2.84±3.91
mCT (ms)	11.5±1.65	67.4±40.6	**10.1**±1.40	10.2±1.38

F. Discussion

Exhaustive experiments and ablation study are conducted to prove that CNS achieves:

High precision: CNS achieves <0.3° and sub-millimeter precision in real-world experiments. It is much robust to error correspondence than IBVS (Table I, column "Scene-Easy, w/o RANSAC"). It also gains higher precision than end-to-end learning methods [11], [13] (Fig.5a). Ablation study on network architecture (section V-E) verifies the effectiveness of clustering on servo precision and convergence.

Large convergence basin: CNS successfully converges even with large initial pose deviation (Fig.5b, section "L"). It achieves higher success ratio than IBVS in real-world experiments (Table I) and end-to-end learning methods [11], [13] in simulation (Fig.5a). Ablation study on network architecture (section V-E) verifies the effectiveness of proposed GConvGRU on servo precision and convergence.

Generalization: Real-world experiments are conducted with model trained purely in simulation. The distance decoupled velocity design allows CNS workable to scenes of any scale (Supplementary Material, section VII-G) at the cost of estimating an extra scalar. Ablation study on imprecise distance prior estimation (Supplementary Material, section VII-H) shows the tolerance is relatively large. We have also proposed several randomization techniques (Supplementary Material, VII-F) in data generation which can further improve model's robustness to error and generalization.

VI. Conclusion

We present CNS that encodes keypoints correspondence into a graph and empoly GNN as neural policy to achieve generalizable visual servoing with high precision and large convergence basin. Even keypoints are intermittent or partially mismatched, CNS is less dependent on the quality of correspondence and achieves higher convergence and precision than conventional methods [14]. When compared with recently popular learning based methods [11], [13], CNS has better generalization performance because the explicit guidance suppresses the correlation between image appearance and the policy. The training of CNS is independent of specific scenes and can be completely finished in simulation with randomized 3D points. CNS can be directly transferred to real-world scenes without any fine-tuning. The servo of real scenes achieves <0.3° and sub-millimeter servo precision and runs in real-time, which would be a feasible solution to general high precision image servoing tasks.

REFERENCES

[1] 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.

[2] H. Zhong, Z. Miao, Y. Wang, J. Mao, L. Li, H. Zhang, Y. Chen, and R. Fierro, "A practical visual servo control for aerial manipulation using a spherical projection model," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 12, pp. 10 564–10 574, 2019.

[3] 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.

[4] S. Paradis, M. Hwang, B. Thananjeyan, J. Ichnowski, D. Seita, D. Fer, T. Low, J. E. Gonzalez, and K. Goldberg, "Intermittent visual servoing: Efficiently learning policies robust to instrument changes for high-precision surgical manipulation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7166–7173.

[5] O. Kermorgant and F. Chaumette, "Combining ibvs and pbvs to ensure the visibility constraint," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 2849–2854.

[6] 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, 2022.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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)*. IEEE, 2018, pp. 3307–3314.

[12] 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.

[13] 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.

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

[15] 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.

[16] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The confluence of vision and control*. Springer, 2007, pp. 66–78.

[17] 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.

[18] E. Malis, F. Chaumette, and S. Boudet, "2 1/2 d visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 238–250, 1999.

[19] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 224–236.

[20] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4938–4947.

[21] J. Ni, Y. Li, Z. Huang, H. Li, H. Bao, Z. Cui, and G. Zhang, "Pats: Patch area transportation with subdivision for local feature matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 776–17 786.

[22] H. Yu, A. Chen, K. Xu, Z. Zhou, W. Jing, Y. Wang, and R. Xiong, "A hyper-network based end-to-end visual servoing with arbitrary desired poses," *arXiv preprint arXiv:2304.08952*, 2023.

[23] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[24] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[25] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *Acm Transactions On Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.

[26] L. Landrieu and M. Simonovsky, "Large-scale point cloud semantic segmentation with superpoint graphs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4558–4567.

[27] L. Wang, Y. Huang, Y. Hou, S. Zhang, and J. Shan, "Graph attention convolution for point cloud semantic segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 10 296–10 305.

[28] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 16 259–16 268.

[29] T. Cai, S. Luo, K. Xu, D. He, T.-y. Liu, and L. Wang, "Graphnorm: A principled approach to accelerating graph neural network training," in *International Conference on Machine Learning*. PMLR, 2021, pp. 1204–1215.

[30] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[31] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and 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.

[32] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[33] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 2020, pp. 402–419.