2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Learning View and Target Invariant Visual Servoing for Navigation

Yimeng Li and Jana Košecka

***Abstract*— The advances in deep reinforcement learning recently revived interest in data-driven learning based approaches to navigation. In this paper we propose to learn viewpoint invariant and target invariant visual servoing for local mobile robot navigation; given an initial view and the goal view or an image of a target, we train deep convolutional network controller to reach the desired goal. We present a new architecture for this task which rests on the ability of establishing correspondences between the initial and goal view and novel reward structure motivated by the traditional feedback control error. The advantage of the proposed model is that it does not require calibration and depth information and achieves robust visual servoing in a variety of environments and targets without any parameter fine tuning. We present comprehensive evaluation of the approach and comparison with other deep learning architectures as well as classical visual servoing methods in visually realistic simulation environment [1]. The presented model overcomes the brittleness of classical visual servoing based methods and achieves significantly higher generalization capability compared to the previous learning approaches.**

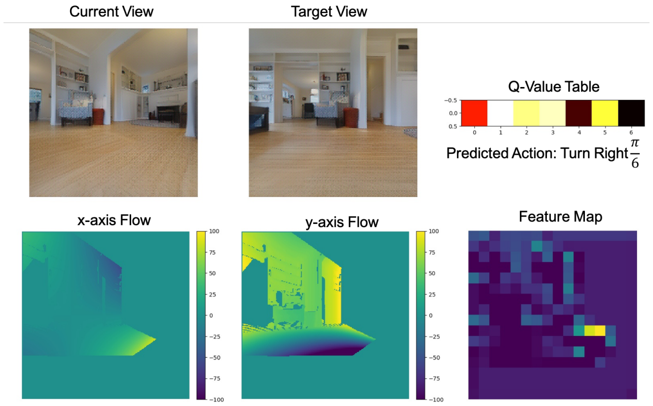

Fig. 1. Visualization of the proposed method. The three images on the top are current view, target view and Q-value table. Brightest color shows up in the left block of the Q-value table indicates taking a turning-right action. The three images at bottom are the visualizations of dense correspondence map on x/y-axis and the feature map output by the perception module.

I. INTRODUCTION

Traditional approaches to navigation in novel environments often required solution to many components, including mapping, motion planning and low level control. The reliance of motion planning on high quality geometric maps and trajectory following on perfect localization, resulted in fragmented and brittle solutions which had to be fine-tuned for particular environments. In contrast to these methods biological systems have more flexible representations of environments and control policies which enable them to robustly navigate in previously unseen environments. These observations and the emergence of effective data driven techniques for learning control policies directly from observations recently spurred large body of research in learning navigation. In this paper we propose a learning approach to visual servoing for mobile robots in indoors environments. In the broader context of navigation task, visual servoing discussed here can be viewed as local navigation skill for view based navigation, where the goal is to reach the desired view or navigate towards the target of interest. While there is a large body of work on classical approaches to visual servoing, they rely on the extraction, tracking and matching of a set of visual features which are difficult and brittle tasks. Related attempts to overcome these difficulties using learning based approaches have been recently considered in [2], with the focus on improving the pose estimation and perception component for 6-DOF pose based visual servoing in table top environments. We study the problem in mobile robot navigation setting and propose to learn the control policy jointly with the perception component. The contributions of this work are as follows:

(i) We show how general image correspondence map can be distilled by deep convolutional networks and trained in an end-to-end manner to learn a policy for target reaching and goal view reaching task; (ii) We design a novel dense reward structure and train the model using deep reinforcement learning (DRL) framework; (iii) We present a comprehensive comparison of our model with alternative deep CNN architectures and training approaches proposed for this task as well as classical image based visual servoing techniques, showing superior performance of our approach. Comprehensive evaluation is carried out using visually realistic simulated household environments [1] with variety of targets and goals, demonstrating good generalization ability of the approach.

II. RELATED WORK

Here we review the related work focusing on local navigation skills most relevant to our approach. A class of local navigation methods assumes that the goal is specified in an agent's local coordinate frame, often assuming perfect localization. Authors in [3] learn a local navigation policy using deep reinforcement learning and Value iteration networks, [4] learn how to predict the next waypoint given a long-range goal and use traditional optimal control to compute the desired trajectory given local ego-centric map of the environment. Kumar et al. [5] consider visual-teach-and-repeat approach where the environment is represented

George Mason University, Computer Science Department `[yli44, kosecka]@gmu.edu`

978-1-7281-7395-5/20/$31.00 ©2020 IEEE

in terms of trajectories experienced in exploration. Efficient retrieval of the views along with the actions enables novel traversals of the environment.

Locomotion policies proposed in [6] and [7] use siamese network to extract feature maps from the input images and estimate the discrete motion. Pathak et al. [8] combines forward dynamics with inverse dynamics model to solve the bi-modal ego-motion estimation problem. Savinov et al. [9] stack the start and goal image up as an input to their locomotion network, training the model in a supervised way. Disadvantage of these models is poor generalization capability and very dense sampling of the intermediate views to represent the trajectories.

Related problem of semantic target driven navigation was considered by [10], [11] where object is specified as an image or as an semantic category [12] considering mid-range navigation tasks and more loose definition of goal success. Similarly to us many of the learning based methods train their models in simulated environments, followed by transfer or adaptation of the learned policies to real robots. Authors in [13] and [3] successfully transfer their model trained on GibsonEnv [1] to the real world.

Besides going to a semantic goal, people have also explored other goal representations; Amini et al. [14] use a local topological map and Codevilla et al. [15] steer a toy truck through high-level map related commands.

There is also a large body of work on classical visual servoing methods. More recent discussion can be found in [16]. Image based visual servoing has been adapted for short range mobile robots navigation in [17] and pose based visual servoing task was studied in [2] where 6-DOF pose regression is estimated by deep networks followed by traditional control.

The work closest to ours is Sadeghi et al. [10], [18] and Yexin et al. [19]. They both study the target reaching problem and use DRL to train the model in simulation. In [10] the policy is guided by heatmap obtained through the correlation in the feature spaces of object and current robot's view. The attention mask obtained from semantic segmentation is computed by [19] and used as an input for the policy learning. Both approaches focus on the problem of reaching *semantic target* and rest on the availability of powerful architectures and representations pre-trained for object detection and semantic segmentation. Furthermore the attention mechanism does not lend itself to *homing* scenarios in navigation, where the goal of the agent is specified as the goal view and may not contain interesting objects.

III. APPROACH

Consider an agent that operates in a novel indoor environment. Instead of navigating with the help of a pre-built metric map, our agent is provided with a set of images taken at different locations, forming a view based topological map of the environment [9]. Our goal is to train the agent to do short-range navigation to reach desired location or target of interest in the field of view of the agent. Let's assume that the agent starts at some random state s_0 and obtains an observation I_0. State s is characterized by the pose of the agent x, y coordinates and the heading angle θ in a world reference frame unknown to the agent. The observation I can be either an RGB image or an RGB-D image including depth information as additional channel. A target view image I_g taken at goal position s_g is provided to the agent. I_g is assumed to have some overlap with the initial field of view I_0. The goal for the agent is to learn a policy for reaching the goal state s_g by executing a sequence of actions a. The action space A can be either continuous or discrete depending on the control method.

We approach this problem using learned data-driven strategy and compare and discuss the advantages of this approach to classical image based visual servoing [20] and learned visual servoing approaches [2].

Classical visual servoing computes correspondences at few selected points between current observation I_t and target view I_g followed by analytic derivation of the feedback control law for continuous velocities at each each step. Learning method learns a mapping from I_t and I_g to actions in reinforcement learning framework and predicts at each state a discrete action a moving towards the goal location. The action space includes seven actions, which are moving forward $0.1m$ in 7 orientations $[-\frac{\pi}{4}, -\frac{\pi}{6}, -\frac{\pi}{15}, 0, \frac{\pi}{15}, \frac{\pi}{6}, \frac{\pi}{4}]$.

We will start our discussion with the geometry of the problem, discussing the brittleness and drawbacks of the geometric methods to motivate our learning based solution.

A. Classical Visual Servoing

Visual servoing control has been developed in early 90's with the goal of increasing the accuracy of the control of robot end effector by using visual feedback control. Two main classes of systems are position based and image based visual servo. While the position based visual servo strives to first estimate the relative pose between the initial and target view, the image based visual servo derives the error signal directly from measurements. The image features f can vary dramatically (e.g. points or lines), but the effective relationships between robot's pose and its velocities is characterized by feature Jacobian $J_f(s)$ which can be derived analytically

$$\dot{f} = J_f(\mathbf{s})\dot{\mathbf{s}} \tag{1}$$

where $\dot{\mathbf{s}} = [\nu_x, \nu_y, \nu_z, \omega_x, \omega_y, \omega_z]^T$. $J_f(s)$ is also referred to as *interaction matrix*. For image points the relationship between image velocities and motion of the end effector is characterized by the well known optical flow equation [16]. For a holonomic mobile robot moving on the xz-plane, the velocity space is three-dimensional $[\nu_x, \nu_z, \omega_y]^T$ and the optical flow equation reduces to:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} -\frac{\lambda}{Z} & \frac{(u-u_0)}{Z} & -\left(\lambda + \frac{(u-u_0)^2}{\lambda}\right) \\ 0 & \frac{(v-v_0)}{Z} & \frac{-(u-u_0)(v-v_0)}{\lambda} \end{bmatrix} \begin{bmatrix} \nu_x \\ \nu_z \\ \omega_y \end{bmatrix} \tag{2}$$

where $\dot{u}$ and $\dot{v}$ is the optical flow, λ is the focal length for the camera, Z is the depth of the feature point and (u_0, v_0) is the image principal point. In practice, if $N >= 2$

correspondences are detected, we can stack up rows J_f to get the interaction matrix for all features J. The control law can then be obtained by computing the pseudo-inverse of Eq. (2) for the desired camera motion $\dot{s}^* = J_f^+(f^* - f)$.

For image based visual servoing (IBVS), in addition to challenges of detection, matching and tracking of geometric features, there are additional well-known difficulties for visual navigation tasks [16]. It is possible to have an inconsistent set of feature velocities such that no possible camera motion will result in the required image motion. For example, nearly collinear features will cause very small camera motion. The performance of the model relies on the feature depth Z in Equation (2). Some approaches considered using a constant depth for all the feature points or depth estimates from motion is also helpful. The motion based methods often failed when camera motion got smaller. For the configurations where the desired orientation is notably different from the current orientation and differential drive robots with non-holonomic constraints, the overlap between current view and goal view often becomes too small resulting in the failure of the correspondence computation. The proposed learning based approach helps to overcome the brittleness of the traditional methods.

B. Learned Visual Servoing

We formulate the problem of learning visual servoing (LVS) in the reinforcement learning framework. An agent starts at a state s_0 in the environment and the goal is to navigate to the given target view. At each s_t, the agent receives an observation I_t. Given the observation, the agent takes action a_t and receives a reward $r(s_t, a_t)$. We are interested in learning control policy $a_t = \phi_\theta(\psi(I_t, I_g))$ and representation of current view and goal view, that predicts the action advancing the agent towards the goal, where ϕ_θ is the action module and ψ is the perception module. In traditional Q-learning approach, the goal is to learn a Q-function $Q(s_t, a_t)$ quantifying the goodness of different actions in each state. By following a policy π_θ, the agent produces a sequence of Q-states $(s_t, a_t)_{t=0}^T$ after T steps. Our goal is to train a policy π_θ that maximizes the total reward received through the trajectory.

Deep Q Network: For large state spaces the Q function cannot be computed exactly. We use a Deep Q Network (DQN) [21] to approximate the value of a Q-state, $Q(s_t, a_t)$, the action-value of unobserved pose s_t of the mobile agent. Our DQN architecture has two components: a perception module and an action module as shown in Figure 2. The perception module consists of four convolutional layers, all using ReLU activation and batch normalization for efficient training. The first three convolutional layers have kernel size $5 \times 5 \times 16, 3 \times 3 \times 32, 3 \times 3 \times 64$, strides of $2, 1, 1$ and each of them is followed by a max pooling layer to reduce spatial support of the feature map. The last CNN layer is a 1×1 convolutional layer which outputs a $16 \times 16 \times 1$-sized feature map. As input we examine several intermediate representations of RGB views before passing them into the perception module. For the action component the feature map is flattened into a 256-dimensional vector followed by a fully-connected layer which outputs Q-values for all actions in our feedback policy (see Figure 1). Note that our model's architecture is highly flexible. Given different kinds of visual input, we can vary the design of the perception module and leave the action module untouched. We also tried to insert an LSTM layer in front of the FC-layer to learn a recurrent policy. We didn't get a performance boost compared to the reactive policy. All the experiment results mentioned in the paper are achieved without using an LSTM layer.

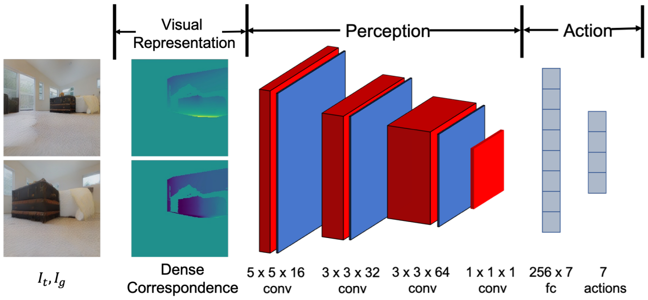

Fig. 2. Learned Visual Servoing Architecture. Visual representation is the dense correspondence extracted from two input images I_t, I_g. Perception module extracts features from the visual representation. Action module predicts the Q-values for each action based on the feature input.

Input Visual Representations: Motivated by classical visual servoing, the input to our perception module is the map of dense correspondences. In the first stage we compute the dense correspondences (shown in Figure 2) using ground truth information of known pose and camera depth. Learned and fine-tuned dense correspondences as in [22] can be later incorporated in our model. The input is a two-channel image where each channel represents the relative offset on x/y-axis from each point in the current view to a corresponding point in the target view. We also observe that applying image smoothing to the dense correspondence map during testing stage improves the model's stability, as smoothed correspondence map has fewer depth discontinuities that can negatively affecting the final prediction. In experiments (Exp D) we corrupt the high-quality dense correspondences in various ways to see the effect of the noise on the final robustness of our model.

Reward: For the task of driving to a target view, the robot traverses a trajectory so as to minimize the error between start view and goal view. Since view difference between two different locations with similar orientation is rather small, the image error is a weak signal for our learning task. To measure the 'progress' to the goal during navigation, we compute d_{polar} distance between poses (x_t, y_t, θ_t) and (x_g, y_g, θ_g). d_{polar} is expressed in the coordinate frame of the goal, where ρ is the distance between robot's center and the goal position, α is the angle between robot's reference frame and the vector connecting center of the robot with the goal position and β is the angle difference between current orientation of the robot and heading direction. The distance to the goal d is a weighted sum of these position and angular distances. Note that when at the goal all $\rho = \alpha = \beta = 0$. The distance to the goal is related to feedback control law error for differential drive robots outlined in [16].

$$\begin{aligned}\rho &= \sqrt{(x_g - x_t)^2 + (y_g - y_t)^2} \\ \alpha &= \arctan(y_g - y_t, x_g - x_t) - \theta_t \\ \beta &= \theta_g - \arctan(y_g - y_t, x_g - x_t) \\ d_{polar} &= \rho + \lambda_\alpha \rho(\alpha, 0) + \lambda_\beta \rho(\beta, 0)\end{aligned} \tag{3}$$

d_{polar} distance enforces the agent to travel towards the target and rotate to align with target pose when the agent is in the vicinity of the target. In practice, λ_α and λ_β are both set to 0.2. We also consider d_{pose} distance [23] for comparison and use d for clarity to denote the distance below.

The reward is designed to encourage the robot to move in the direction to minimize the distance to the goal. The reward function is,

$$R = \max\left(0, \min\left(\frac{d_{t-1}}{d_{init}}, 1\right) - \min\left(\frac{d_t}{d_{init}}, 1\right)\right) \tag{4}$$

where d_{init} is the initial distance between the agent and the goal and d_t is the current distance of the agent to the goal. The right term in the $\max$ function measures the progress of the agent towards the goal after taking action a_t.

Note that we are having a dense reward setup as the environment is simulated. We leave the design of sparse rewards to future work when we train an agent in real-world scenes.

IV. EXPERIMENTS

Datasets: All the experiments are completed in GibsonEnv [1]. Gibson environment contains visually realistic reconstructions of indoors scenes, with varying appearances and layouts. We randomly select 16 indoor scenes as training data and 6 others for testing. During training, we randomly sample the starting robot poses. Examples of sampling on one test scene is shown by Figure 3. Sampled initial views and target views are shown by Figure 4. The target views are chosen at certain distance and orientations away from the starting point. The distance ranges from 0.5 to $4.0m$. The orientation and target pose heading angle ranges from $-\pi/4$ to $\pi/4$ with respect to the initial pose. Usually it will take the agent from 5 to 35 steps to reach the goal. We make sure that the target view and start view has at least 16×16 pixel overlapping area. During testing, for each scene, we manually select approximately 10 starting poses where there is enough open space in front of them. The target images are chosen in a similar way as training data. We evaluate different approaches by computing the success rate of visual servoing results. The success criteria is that the robot pose is within $0.2m$ to the target pose during navigation.

Training Details: We use Actor-Critic architecture [21] and replay memory when training our model. Each minibatch consists of losses over 128 random state-action tuples sampled from replay memory. Each tuple contains the state representation, the action and the reward after taking the action. We use RMSprop Optimizer, learning rate of 10^{-5}. The model is optimized for 50000 iterations on a single GPU, which takes 6 hours.

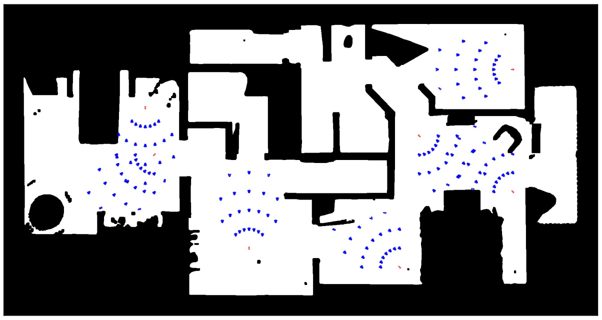

Fig. 3. Occupancy map of the 'Dardan' scene in the testing dataset. We sample 7 starting viewpoints in the environment represented by the red triangles. The blue triangles are the target poses. Their distance to the start viewpoint and orientation vary from each other.

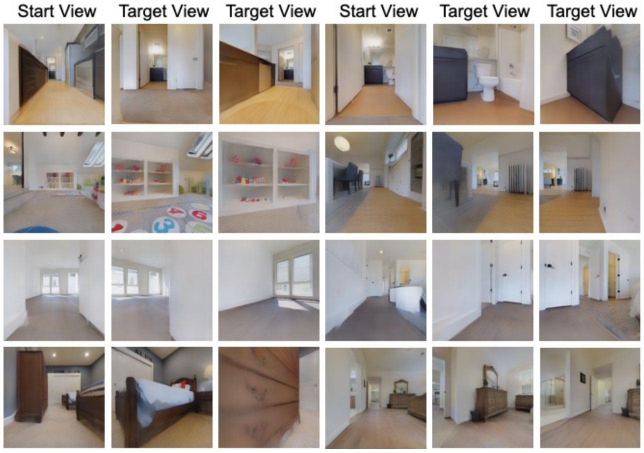

Fig. 4. Visualizations of start views and target images from different test scenes. First and fourth column shows the start views while second, third, fifth and sixth column shows the target views.

A. Intermediate Representations for LVS

We compare the performance of different perception module architectures while using the same reward structure and action module in DQN.

LocomotionNet [9]. Input to the network is a triplet of RGB images comprising of previous observation, current observation and target image. The input images are stacked up into a 9-channel image and put through a sequence of convolutional layers to extract features. Convolving the current view with previous view will provide the knowledge of the previous action. Convolving the current view with target image indicates the signal to the goal.

FlowNet. We combine current view and target image into a pair and use FlowNet [24] to compute the optical flow. The optical flow is represented as a two-channel flow map that is an input of our perception module.

SiameseNet. The perception module follows the architecture proposed by [6]. It is modeled as a siamese network for computing feature maps respectively from input images. The current observation and target image are used as inputs. The output feature maps are flattened into a vector and then concatenated into a large feature vector as the final output.

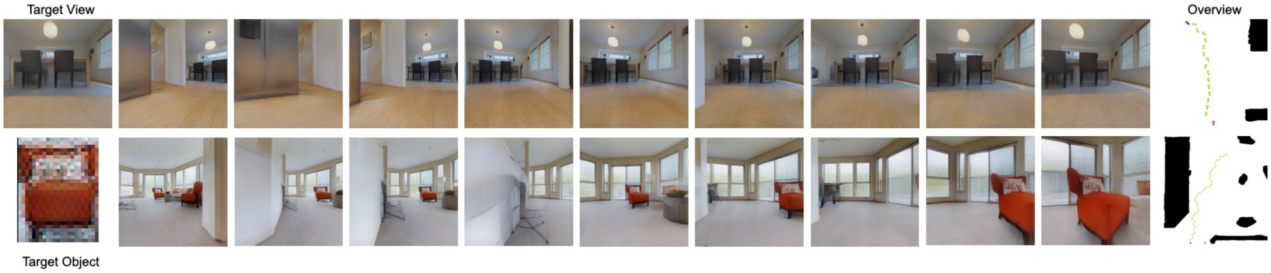

Fig. 5. Examples of a robot driving to a view (first row) and driving to an object (second row). The first column shows the target view and the target object. The last column is the overview of the two trials. Between them are the observations along the driving.

Inverse Dynamics. We adopt the architecture of [8] to learn the network weights and train the model through supervised learning. Then we fix the weights of the trained model and use it as the perception module.

Correlation Map [25]. The perception module computes the correlation between patches from two feature maps and outputs the correlation map as the final visual representation. The feature map extraction architecture is similar to our perception module but having two more pooling layers. We compute the correlation between two $8 \times 8 \times 64$ feature maps and obtain a $8 \times 8 \times 8 \times 8$ correlation map.

We separate the test cases into short-range navigation where the target location is within 10 steps starting from the initial pose and longer-range navigation where the agent needs at least 15 steps to reach the goal. It takes the robot more than 25 steps to reach the furthest goal.

TABLE I
VISUAL REPRESENTATION

Approach	Short/Long-Range
LocomotionNet [9]	26.5% / 18.2%
SiameseNet [6]	14.3% / 28.8%
CorrespondenceMap	85.7% / 80.3%
Smoothed CorrespondenceMap	**90.9%** / **86.9%**

Table I shows the experiment results. We didn't get satisfying results for some of the approaches so their results are missing from the table. The optical flow detected by FlowNet is not very helpful because FlowNet is trained to learn small displacement while our displacement spans over 80 pixels in some test cases. In the following experiment, we use smoothed correspondence map as the default input when we evaluate our LVS model.

B. Reward

Here we hold up the input visual representation to be the dense correspondence map, but vary the reward structure. The dense correspondence which we denote as ground-truth correspondence is computed for each pixel in the common field of view using the available depth information. We compare the performance of using two distance metric, d_{polar} distance and d_{pose} distance [23], and two reward setups, distance minimization reward $DistMinimize$ used by our model and Sadeghi's progressive reward $Progress$ [10].

TABLE II
REWARD STRUCTURE

Reward	Distance Metric	Success Rate
DistMinimize	d_{polar}	**83.5%**
Progress	d_{polar}	73.2%
Progress	d_{pose}	54.9%

Table II shows the results. Using both d_{pose} metric and using $DistMinimize$ reward boosts the success rate for more than 10%. The advantage of $DistMinimize$ over $Progress$ is that in the latter, the robot will receive a positive reward even though the distance to the goal is not shortened. This makes the agent hesitant to move towards the goal. Furthermore, using d_{pose} distance forces the robot to head towards the goal and avoids the risk of losing correspondence (overlap) to the target view. This is helpful for some difficult test case as shown in Figure 6.

C. Classical Visual Servoing Evaluation

Here we evaluate IBVS on a non-holonomic robot from two aspects: image point features and depth data. We experiment with three point feature variations: ground-truth sparse correspondence (GtCorresp), learned sparse correspondence (LearnedCorresp) and SIFT. For ground-truth correspondence, we deliberately select 4 correspondences with the largest offset, as these are the feature points most indicative of the motion between images. We don't evaluate dense correspondence since the interaction matrix inverse computation is time-consuming and it is more likely that the feature velocities are inconsistent when you have a large number of features. We take an off-shelf learned correspondence [26] method as input. Falsely detected correspondence are removed through geometric verification, both in case of learned correspondences and SIFT features. We try four depth variations: ground-truth depth (GtDepth), constant depth (ConstantDepth), noisy depth (NoisyDepth) and no depth. Ground-truth depth image is taken directly form the simulated environment. Constant depth is setup to be 4.0 as all the target viewpoints are within 4 meters. Noisy depth is computed by adding a gaussian noise with $\mu = 0, \sigma = 0.5$ to the ground-truth depth image. For the no depth setup, we only compute the angular velocity ω_y and use constant velocity $\nu = 0.1$.

TABLE III

DIFFERENT IBVS SETUPS

Feature+Depth	Textured	Nontextured	Corridor
GtCorr+GtDepth	80.4%	72.2%	80.4%
LearnedCorr + GtDepth	40.0%	23.2%	32.9%
SIFT + GtDepth	41.3%	18.5%	24.7%
GtCorr + noisyDepth	52.0%	42.5%	46.8%
GtCorr + constantDepth	57.0%	56.2%	48.3%
GtCorr + noDepth	32.8%	29.9%	28.2%
GtDenseCorr + noDepth	**87.6%**	**88.8%**	**84.8%**

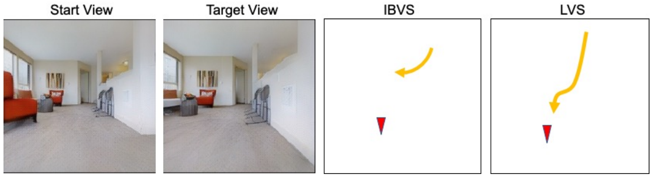

Fig. 6. Visualization of a test case where correspondence is missing during navigation. Given the start view and goal view, the robot should move right then turn left. Using IBVS, robot stops due to loss of FOV overlap. Using LVS, robot is able to reach the goal.

Table III, compares the success rate of different IBVS methods under various environments. The results show that classical visual servoing's performance is highly correlated with the quality of feature points and depth data. Using carefully selected ground-truth correspondences boosts the model's performance by more than 40% than using detected features. Using constant depth instead of ground-truth depth degrades the performance by 20%. Using noisy depth results in 28% drop in success rate demonstrating that IBVS is sensitive to the depth changes. We also evaluate our LVS method (GtDenseCorr + noDepth) on the same test environment. It outperforms IBVS by at least 8% in textured and nontextured environments especially in the following case: when the target location is on the opposite of the robot's orientation, the robot might lose the common area between current observation and the goal image. In another word, there is no correspondence between I_t and I_g. A robot using IBVS will stop under this condition while it will continue move forward using LVS based on its learning experience. Figure 6 demonstrates this hard case.

D. Noisy Correspondences

To evaluate the robustness of our trained model we add gaussian noise to the displacement at each pixel in the correspondence map. The variance σ of the gaussian distribution controls the amount of offset. We also vary the density of the detected features using uniform distribution. Parameter $Coverage$ controls the probability of a pixel having a correspondence. Before we input the noisy correspondence map to the model, we convolve the map with averaging smoothing filter.

Figure 7 shows the results. Our model is robust to large offset errors and still achieves 80% success rate when the offset deviates from the ground-truth for up to 32 pixels. For the missed correspondence noise, it is surprising that the performance is still above 75% when 50% of features is missing. The smoothing reduces the sparsity and has favorable effect on the resulting policy. This demonstrates that our model is robust to several types of correspondence errors.

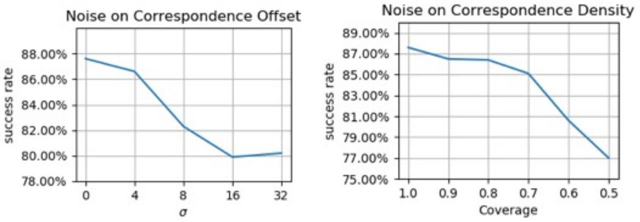

Fig. 7. Comparison of the performance of the model under different noise settings. Left image demonstrates the Gaussian noise on correspondence offset. Right image demonstrates the uniform noise on correspondence coverage

E. Target Object Servoing

Even though our model is not trained on object targets, it can be easily adapted to navigate towards semantic targets. To identify the target of interest, we run an object detector [27] on the images collected by randomly driving the robot in the environment and crop the detected object of interest. The crop is then used as the target image, followed by computation of the correspondence map between current view and target view. The correspondence map is input to the model and we drive the robot by following the action predictions. Figure 5 gives out one example of a robot driving to an object. This suggests that the proposed method might be suitable for semantic target navigation considered previously [10]. We don't do large-scale quantitative evaluations of our model's performance on the driving-to-object task as the object locations are not labeled in the environment.

V. CONCLUSION

We demonstrated learning view and target invariant visual servoing for local navigation. We examine the performance of a variety of input representations and train the model using deep Q-learning framework. Both the input of our model and the reward structure are motivated by classical visual servoing methods. The ability to train the model in an end-to-end fashion significantly improves the robustness and performance of the approach compared to classical visual servoing methods, where a small set of fixed features is selected for the task. We present comprehensive comparison of the model to alternative representations proposed in the literature. While the current model assumes dense correspondence, it is robust to errors in the correspondence maps. In the future work we plan to incorporate depth information explicitly, carry our more extensive experiments with target objects and presence of obstacles and validate our approach on mobile robot platform.

REFERENCES

[1] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: Real-world perception for embodied agents," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9068–9079.

[2] Q. Bateux, E. Marchand, J. Leitner, F. Chaumette, and P. Corke, "Visual servoing from deep neural networks," *arXiv preprint arXiv:1705.08940*, 2017.

[3] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2616–2625.

[4] S. Bansal, V. Tolani, S. Gupta, J. Malik, and C. Tomlin, "Combining optimal control and learning for visual navigation in novel environments," *arXiv preprint arXiv:1903.02531*, 2019.

[5] A. Kumar, S. Gupta, D. Fouhey, S. Levine, and J. Malik, "Visual memory for robust path following," in *Advances in Neural Information Processing Systems*, 2018, pp. 765–774.

[6] P. Agrawal, J. Carreira, and J. Malik, "Learning to see by moving," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 37–45.

[7] D. Jayaraman and K. Grauman, "Learning image representations tied to egomotion from unlabeled video," *International Journal of Computer Vision*, vol. 125, no. 1-3, pp. 136–161, 2017.

[8] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell, "Zero-shot visual imitation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 2050–2053.

[9] N. Savinov, A. Dosovitskiy, and V. Koltun, "Semi-parametric topological memory for navigation," *arXiv preprint arXiv:1803.00653*, 2018.

[10] F. Sadeghi, "Divis: Domain invariant visual servoing for collision-free goal reaching," *arXiv preprint arXiv:1902.05947*, 2019.

[11] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3357–3364.

[12] A. Mousavian, A. Toshev, M. Fišer, J. Košecká, A. Wahid, and J. Davidson, "Visual representations for semantic target driven navigation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8846–8852.

[13] X. Meng, N. Ratliff, Y. Xiang, and D. Fox, "Neural autonomous navigation with riemannian motion policy," *arXiv preprint arXiv:1904.01762*, 2019.

[14] A. Amini, G. Rosman, S. Karaman, and D. Rus, "Variational end-to-end navigation and localization," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8958–8964.

[15] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–9.

[16] P. Corke, *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised*. Springer, 2017, vol. 118.

[17] S. R. Bista, "Indoor navigation of mobile robots based on visual memory and image-based visual servoing," Ph.D. dissertation, 2016.

[18] F. Sadeghi, A. Toshev, E. Jang, and S. Levine, "Sim2real viewpoint invariant visual servoing by recurrent control," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4691–4699.

[19] X. Ye, Z. Lin, J.-Y. Lee, J. Zhang, S. Zheng, and Y. Yang, "Gaple: Generalizable approaching policy learning for robotic object searching in indoor environment," *IEEE Robotics and Automation Letters*, 2019.

[20] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo based control," *IEEE Transactions on Robotics and Automation*, 2018.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[22] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker, "Universal correspondence network," in *Advances in Neural Information Processing Systems*, 2016, pp. 2414–2422.

[23] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[24] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2758–2766.

[25] I. Rocco, R. Arandjelović, and J. Sivic, "End-to-end weakly-supervised semantic alignment," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6917–6925.

[26] 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.

[27] F. Massa and R. Girshick, "maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch," https://github.com/facebookresearch/maskrcnn-benchmark, 2018.