Exploring Convolutional Networks for End-to-End Visual Servoing

Aseem Saxena*[1], Harit Pandya*[1], Gourav Kumar[1], Ayush Gaud[1] and K. Madhava Krishna[1]

***Abstract*— Present image based visual servoing approaches rely on extracting hand crafted visual features from an image. Choosing the right set of features is important as it directly affects the performance of any approach. Motivated by recent breakthroughs in performance of data driven methods on recognition and localization tasks, we aim to learn visual feature representations suitable for servoing tasks in unstructured and unknown environments. In this paper, we present an end-to-end learning based approach for visual servoing in diverse scenes where the knowledge of camera parameters and scene geometry is not available a priori. This is achieved by training a convolutional neural network over color images with synchronised camera poses. Through experiments performed in simulation and on a quadrotor, we demonstrate the efficacy and robustness of our approach for a wide range of camera poses in both indoor as well as outdoor environments.**

I. INTRODUCTION

Visual servoing (VS) refers to the control of robot motion using data from vision sensors. Vision sensor integration enables robotic systems to work outside controlled industrial settings. As a consequence, it has applications in diverse areas such as robotic surgery, autonomous navigation and manipulation for household robots. The objective is to move the robot in Cartesian space from an arbitrary starting pose (location and orientation) to a fixed goal pose. This is achieved by iteratively minimizing the error between the current and goal pose.

Position Based Visual Servoing (PBVS) defines the error in Euclidean space, which results in a simpler control law and minimal length trajectory [1]. However, PBVS requires a 3D model of the scene and camera parameters to be known before hand, which is a major bottleneck in practical implementation of PBVS methods. Image based visual servoing (IBVS), on the other hand, describes the error function in image space by extracting a set of visual features. IBVS controller attempts to move the robot in such a way that the visual features attain the desired configuration i.e. it minimises the error in image space. This requires a mapping of the feature velocity in image space to robot motion in Euclidean space via the analytical computation of image Jacobian that leads to various issues such as attaining local minima, exceeding joint limit and so forth [2]. Another issue with IBVS approaches is the extraction of unambiguous features that truly represent the pose information, which is a non-trivial task. Classical IBVS approaches use geometrical primitives like locations of points, lines, contours etc. as visual features [1]. However, these methods require an accurate feature matching for convergence. Recent IBVS methods consider appearance based features such as pixel intensities [3], image gradients [4] etc. These methods do not require an explicit matching step, however, the number of features is typically very large that results in a smaller convergence domain.

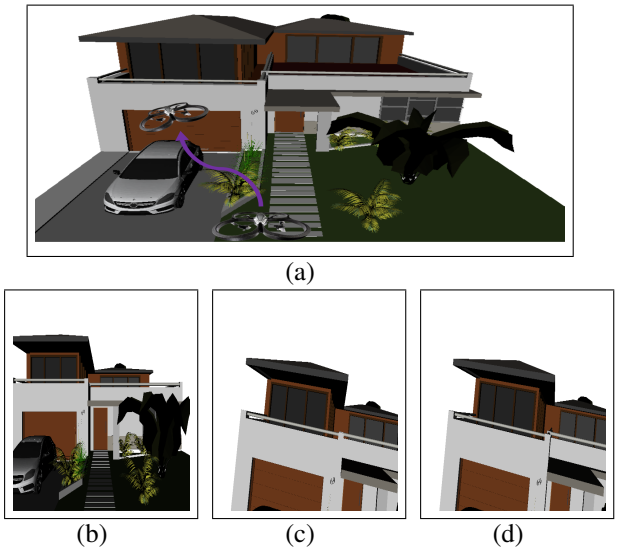


Fig. 1. **Visual servoing using CNN.** We present a learning based approach for end-to-end visual servoing using CNNs i.e. given a desired pose and initial pose in the form of images, our framework predicts visual servoing control commands. (a) A 3D model of an outdoor scene from our dataset with an initial and desired location of the robot. (b) Initial pose. (c) Desired pose. (d) Resultant pose attained by the robot using our framework. Note that our network does not assume prior knowledge of camera parameters and geometry of the scene. Also note the large camera displacement between initial and desired pose.

In this paper, we address the following question - is it possible to learn the motion required to attain a desired pose from an initial pose only from visual feedback? Recent breakthroughs in computer vision suggest that data driven frameworks efficiently learn high-level semantic representations from images, especially for a large number of examples [5]. Motivated by recent advances in machine learning, especially deep learning, we aim to learn an optimal set of image representations that estimate the relative transformation required to attain a desired pose. We explore convolutional neural network (CNN) architectures to learn such transformations in an end-to-end paradigm. Unlike other visual servoing approaches, our framework eliminates the need for extraction and tracking of features. Moreover, prior knowledge about the camera intrinsics and the scene's 3D geometry is not required. Experiments show that our model

*Equal contribution.

[1] International Institute of Information Technology, Hyderabad, India.
{harit.pandya, gourav.kumar}@research.iiit.ac.in
{aseem.bits, ayush.gaud}@gmail.com, mkrishna@iiit.ac.in

Harit Pandya is supported by TCS Reserach PhD fellowship.

also has a large convergence domain across a variety of synthetic and real world scenarios.

In this paper, we present a Convolutional Neural Network trained for performing visual servoing on diverse environments without knowledge of the underlying scene's geometry. We have trained the network on the publicly available 7-Scenes dataset [6] as this dataset provides large variations across scenes and covers a wide range of camera transformations between frames. We evaluate our network on 5 synthetic 3D models using free camera paradigm and on a real world scene using a quadrotor. Our simulation based testing framework allows us to compute ground truth camera transformations that can be used to compute the error of our system's estimates. Figure 1 shows an exemplar of servoing result. Figure 1(a) shows the scene on which servoing was performed. Figure 1(b-d) shows initial pose, desired pose and resultant pose attained after visual servoing. Note that although there is large camera motion between initial and desired pose, the camera still reaches close to the desired pose using our method.

II. Related work

Most of the previous image based visual servoing approaches rely on hand-crafted visual features for representing images. The control law could be seen as gradient descent over the feature error [3]. This requires image Jacobian or interaction matrix to be computed analytically. For several features widely used by modern computer vision techniques, it is difficult to represent analytically, for instance, Histogram of Oriented Gradients (HOG). Another line of approaches intend to numerically estimate the interaction matrix. However, due to high non-linearity in interaction matrix, it is difficult to get an accurate estimation. Also, numerical methods are vulnerable to conditioning and singularity issues. Neural Network based methods have been used for learning interaction matrix but the selection of features was hand-engineered. Readers may refer to [7] for a detailed review of Jacobian learning and estimation methods. Recently, support vector machines have been used to learn pose specific representations for visual servoing across object instances [8]. Again, the interaction matrix was numerically estimated. There has been significant work on reinforcement learning (RL) approaches [9], [10] for end-to-end visual servoing. However, parameters learned by RL are specific to the environment and task, hence it becomes difficult to generalise RL for new environments. On the other hand, our approach is end-to-end i.e. we learn visuomotor representations for direct control. Moreover, our approach generalises well on unknown environments.

Techniques for pose estimation, camera relocalization and visual odometry have been successfully applied in approaching the visual servoing problem. There have been works on absolute scene 6D pose estimation from a single monocular image in the recent past which are data-driven [11], [12]. Kendal' et al. [12] train a CNN to regress the 6D pose of the camera from a single monocular image in real time. Our approach differs from theirs as we wish to learn relative

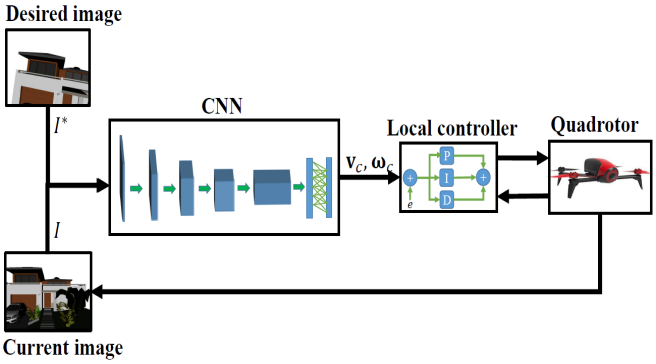


Fig. 2. **Overview of the proposed approach.** Given an image I^* representing desired pose in the image space and the current image measurement I from camera, we use a CNN to estimate relative camera transformation ${}^{c}_{c^*}\mathbf{T}$ required to reach I^* in image space. Considering noise in estimation of ${}^{c}_{c^*}\mathbf{T}$, we take a servoing step of length λ in direction of ${}^{c}_{c^*}\mathbf{T}$.

camera pose from a pair of images. As natural scenes change over time, systems which estimate the absolute pose of a scene are bound to falter as a viewpoint can be remarkably different visually from the same viewpoint at a different time. Rather, we consider the relative pose between two frames to be much more meaningful. Two images with sufficient scene overlap offer more information and context than a single image. Some recent works have approached the problem of camera ego-motion estimation which has applications in visual servoing [13], [14], [15]. Agarawal et al. [13] explore the idea of feature learning using egomotion as ground truth instead of manually annotated labels. They demonstrate camera ego-motion estimation by learning a Siamese Style CNN with two images as input and the relative camera transformation as the ground truth. However, our work significantly differs from theirs in multiple ways. We perform regression over the image pairs whereas they perform classification. Also, we use a different Network architecture, loss function and optimization scheme for our task. Costante et al. [14] train a network to estimate frame to frame visual odometry by taking the optical flow between image pairs as input. Ours does not require the computation of optical flow. To the best of our knowledge, there has been no work which directly addresses the problem of visual servoing by leveraging powerful CNN based image features.

Contributions: Our contributions could be summarised as following. Firstly, we present a CNN based learning framework for visual servoing. Our framework generalises well over a wide range of synthetic and real world scenarios. We rigorously and systematically evaluate our approach in simulation and on real scenarios using a quadrotor. Secondly, as there are no benchmarking datasets for visual servoing and due to the presence of dynamic control, it is not feasible to provide access to pre-capture images similar to most of the available datasets. We would publicly release 3D models of scenes used for testing along with the necessary scripts.

III. Overview

Assuming eye-in-hand configuration and world origin coinciding with the given object's center, we denote the

camera's pose in SE(3) at given time in a fixed global frame as $\mathbf{c}$. Given a scene X and a desired camera pose in the same global frame c^*, the goal of a visual servoing scheme is to find a camera transformation ${}^{c*}_{c}\mathrm{T}$, such that $c^* = {}^{c^*}_{c}\mathrm{T}c$. For image based visual servoing (IBVS), current pose and the desired pose are represented in the form of a set of features extracted from images $\mathbf{s} = \phi(KcX)$ and $\mathbf{s^*} = \phi(Kc^*X)$. Where K is the camera's intrinsic matrix and $\phi(\cdot)$ is the feature selection criterion. For IBVS, the goal is modified to finding the transformation ${}^{c*}_{c}\mathrm{T}$ such that the error in features $\mathbf{e} = \mathbf{s} - \mathbf{s^*}$ is regulated to zero at desired pose. The task is achieved by minimizing $\mathbf{e}$ iteratively and controlling the camera velocity, $\mathbf{v} = -\lambda \mathbf{L_s}^{+}\mathbf{e}$, where $\mathbf{L_s}$ is the interaction matrix that maps the rate of change of features to velocity and $(\cdot)^{+}$ represents pseudoinverse operation as defined in [1]. On the other hand, in position based visual servoing (PBVS), the camera pose at any given time c is inferred from the scene and image measurements. However, inferring camera pose from a single image requires the knowledge of the scene and camera parameters. In this work, we aim to jointly learn the representations $\mathbf{s}$ the describe the image and error $\mathbf{e}$ from a pair of images I and I^* without camera parameters K and geometry of the scene X. Our framework takes RGB images I and I^* as the input and estimates the camera transformation ${}^{c}_{c*}\mathrm{T}$ thereby waving the requirement for any feature computing or matching step. Further, using image based feedback, we estimate the control commands ${}^{c}_{c*}\hat{\mathrm{T}}$ to attain the desired camera pose c^*. Figure 2 shows the overall pipeline of the proposed framework.

IV. Network Architecture

A. FlowNet

Convolutional Neural Networks have recently been shown to perform well on large scale visual recognition tasks [5]. In the recent past, research on training CNNs for per pixel prediction tasks such as optical flow has started to surface [16]. FlowNet by Fischer et al. [16] approaches the problem of optical flow in a supervised learning setting. Optical flow prediction involves both per pixel localization and learning powerful representations. We leverage these aspects of FlowNet to learn camera ego-motion. The motivation behind this effort is that traditionally optical flow has been used to estimate visual odometry [17]. A network which could robustly estimate optical flow would also be able to estimate camera ego-motion since both problems involve correspondences between image pixels. FlowNet is trained to predict optical flow using image pairs as input and their x-y flow fields as ground truth (Figure 3(a)). The images are stacked together to form a 6 channel image which is passed through multiple convolutions and ReLu non-linearities. Convolutional Neural Networks involve down-scaling of feature maps, which is necessary for the training phase to be computationally feasible. As optical flow is a per pixel prediction task, it would require a feature map which is up to scale to predict a flow field of higher resolution. In order to provide dense per-pixel predictions, 'upconvolution' is performed on the coarse feature map to get it up to scale. 'Upconvolution' involves unpooling (bilinear upscaling of the feature map) followed by a convolution (refer Figure 3(b)). Similar layers have been used previously [18]. In this way, information from both coarser and finer feature maps is preserved. Upconvolution is performed at multiple scales which ultimately results in a two channel feature map which is 16 times the resolution of the last coarse feature map and 1/4 times the resolution of the image input. Our network differs slightly from FlowNet as we discard the loss layer of FlowNet and instead feed the final feature map to a fully connected layer with ReLu non linearity and dropout followed by separate regression layers for translation and rotation [12].

V. Training

A. Loss function and Optimization Scheme

Our network takes in two monocular images I, I^* and outputs a pose vector $\mathbf{p}$ comprising of a relative (I^* with respect to I) translation $\mathbf{x}$ and rotation $\mathbf{q}$ in quaternion form.

$$\mathbf{p} = [\mathbf{x}, \mathbf{q}] \tag{1}$$

To regress relative pose, we consider the following objective loss function similar to [12].

$$loss(I, I^*) = \|\hat{x} - x\|_2 + \beta \left\| \hat{q} - \frac{q}{\|q\|} \right\|_2 \tag{2}$$

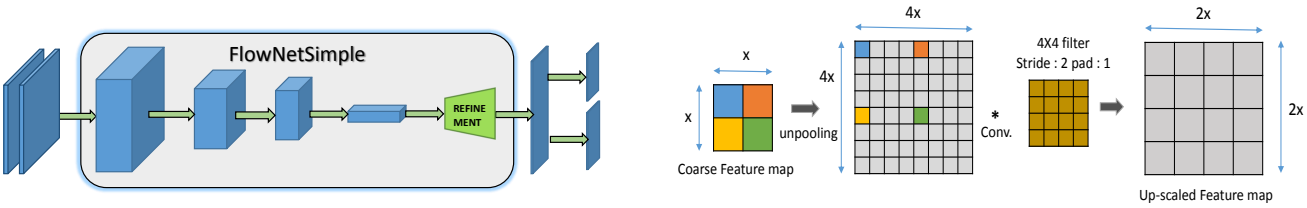
β is chosen so as to keep the expected value of translation and rotation errors to be equal. We found β as $500,000$ to be optimal for training. The motive behind deploying the loss function is that both the translation and rotation regressors are loosely coupled and therefore, are not being denied the information to factor position from orientation and vice versa.

B. Data-Preparation and Implementation Details

We use the Train Split of 7 Scenes Dataset to train our networks. Ground truth is present for each frame in the form of 4×4 homogeneous matrix. For an image frame in a sequence, we take only 10 temporally close frames for computing the ground truth transformation, this is done to ensure that there is partial scene overlap in the two images. Let the absolute pose in homogeneous coordinates of I and I^* be ${}^{O}_{c}\mathrm{T}$ and ${}^{O}_{c^*}\mathrm{T}$ (with respect to some world origin O) respectively, then the Transformation of I^* with respect to I is given by:

$${}^{c}_{c^*}\mathbf{T} = {}^{c}_{O}\,\mathbf{T}\,{}^{O}_{c^*}\mathbf{T} \tag{3}$$

We obtain approximately 500,000 such training image pairs. For training on FlowNet architecture, we resize the images to 512×384 and pass it for training. We use FlowNet's mean subtraction layer to normalize the image data. We use Caffe library [19] to train our networks. The machine has a core i7 processor with 64 GB of RAM. We used a single Titan X GPU to perform training and testing. It took an hour to complete 1000 iterations with a batch size of 40. We perform transfer learning [20], [21] with the official FlowNet model weights released by the authors. This is done in order to get a better network initialization and faster network convergence.



(a) (b)

Fig. 3. **Description of the Network architectures for learning camera motion.** (a) FlowNet – FlowNet takes 2 images stacked together as a six channel image as input and predicts the transformation between the two images. FlowNet consists of Convolutional, ReLu and 'Upconvolution' layers. (b) Upconvolution layer – the coarse feature map is bi-linearly up-sampled to 4 times its size followed by a convolution with a 4×4 filter with stride as 2 and padding as 1. This results in a feature map double the size of the coarse feature map. Upconvolution is performed 4 times at multiple scales.

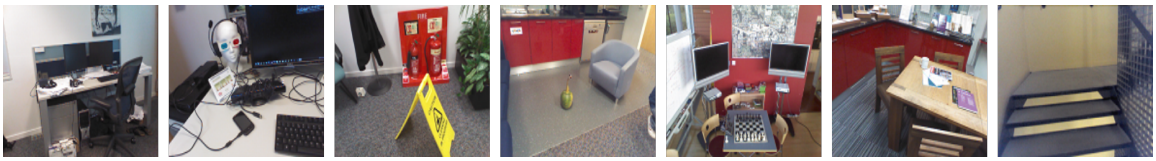


Fig. 4. **7 Scenes dataset** Example images from left to right: Office, Heads, Fire, Pumpkin, Chess, Red Kitchen and Stairs.

We use Adam optimization scheme instead of stochastic gradient descent for minimizing the loss function as it showed faster convergence for training during experiments. We train with base learning rate as 10^{-4} reduced by 50% every $30,000$ iterations. We take momentum, momentum2 parameters of Adam solver to be 0.9 and 0.99 respectively. We use the network weights obtained after $75,000$ iterations of training for all our experiments.

C. Dataset

We train our network on the RGB-D '7 Scenes Dataset' [6]. It comprises of seven scenes of varying spatial extent and clutter as shown in figure 4. The Dataset is challenging due to the presence of image artifacts such as motion blur and reflections. Also, presence of texture-less flat surfaces, sensor noise and varying lighting conditions compound the challenge. We chose this dataset as it comprises of multiple trajectories with a variety of rotational and translational transformations between frames. This would enable us to learn a rich variety of poses with challenging image pairs.

VI. Control Law

The network is trained to compute relative error in pose ${}^{c}_{c^*}\mathrm{T}$ given two images I and I^*. We consider an object centric coordinate system with a frame $\mathcal{F}_{o^*}$ attached to an object. $\mathcal{F}_c$,$\mathcal{F}_{c*}$ denote the current and desired camera frames. In our PBVS scheme, $\mathbf{s} = ({}^{c^*}\mathbf{t}_c, \theta\mathbf{u})$ Consequently, $\mathbf{s^*} = \mathbf{0}$ and $\mathbf{e} = \mathbf{s}$. This formulation results in a decoupling of rotational and translational motions and a simple control law as follows:

$$\begin{aligned} \mathbf{v}_c &= -\lambda {}^{c^*}\mathbf{R}_c^T {}^{c^*}\mathbf{t}_c \\ \omega_c &= -\lambda\theta\mathbf{u}. \end{aligned} \tag{4}$$

Where, ${}^{c^*}\mathbf{R}_c^T$ and ${}^{c^*}\mathbf{t}_c$ are the relative rotation and translation of camera's desired pose with respect to camera's initial pose in frame $\mathcal{F}_c$. ${}^{c}\mathbf{R}_{c^*}^T$ and ${}^{c}\mathbf{t}_{c^*}$ are predicted by our network, given I and I^*. λ is the step size for rotational and translational velocities.

VII. Experiments and Results

We evaluate our approach on non-planar scenes. Since there is no publicly available dataset that allows us to render an entire scene from a given viewpoint, we introduce a new synthetic dataset VSSD consisting of 5 detailed CAD models of various scenes. We use the OpenRAVE simulation framework [22] for rendering scenes since it allows us to quantitatively measure the performance of our approach as the desired camera pose is known in the world frame. Thirdly, we qualitatively evaluate the performance of our approach on our dataset for various initial and desired poses. Finally, we execute our approach in a real environment using a quadrotor. Note that for all the experiments we do not assume any knowledge of camera parameters or depth information of the scene. Another fact worth noting is that the images used in evaluation were not encountered during training of the CNN Model. For simulation experiments we consider free-flying camera model. All the experiments reported here are performed on a system with Intel i7 processor and 64 GB RAM and a single 12 GB Titan X GPU. On this system our approach takes 20s for initialization i.e. loading the network into GPU and henceforth every iteration takes 20ms of which, majority of the time is consumed in forward pass of the network.

A. Visual Servoing Scene Dataset

There are several publicly available datasets for tracking and localization [23], [6]. However, for visual servoing it is difficult to release such a dataset, as it requires image based measurements of the environment where viewpoint changes dynamically. We address this issue by using synthetic 3D models. In the recent past, 3D models have been used by the computer graphics and vision communities to produce large amounts of synthetic data which enable better generalisation

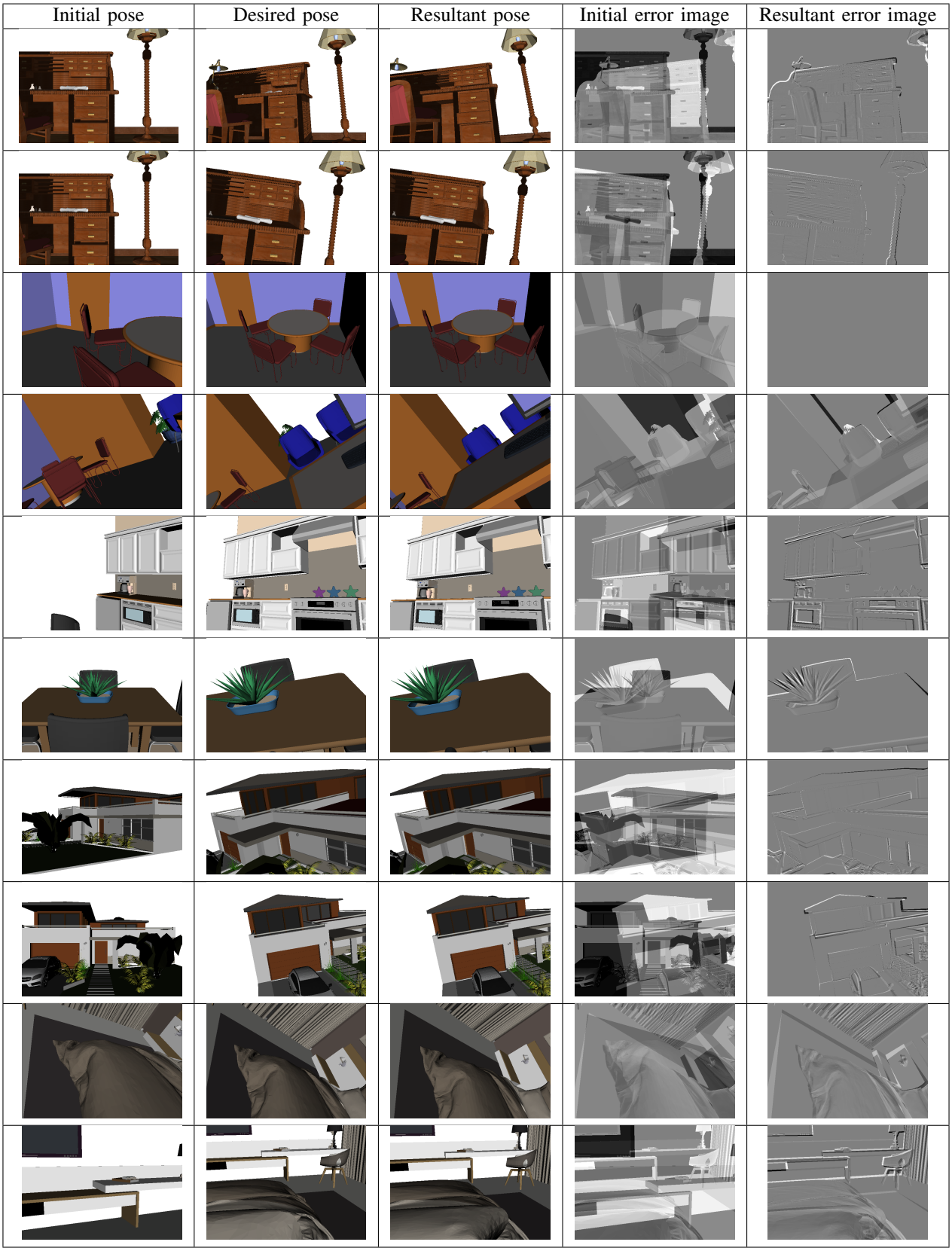


Fig. 5. **Qualitative results.** (a) Initial pose captured by the robot with a random camera pose for provided 3D object from the dataset. (b) Desired pose. (c) Resultant positioning of camera achieved by CNN based visual servoing. (d) Resultant error image. Note the similarity in the resultant pose achieved by the proposed approach compared to the desired pose provided, over a wide range of desired poses.

for deep learning models [24]. However, these datasets are limited to shapes and objects. Recently Handa et al. [25] released a synthetic dataset for scenes. However, the scenes provided are purely depth-based, which makes it unsuitable for visual servoing purposes. 3D positioning is performed for physical objects which limits the scope of reproducing the results for benchmarking purposes.

For this work, we have generated Visual Servoing Scene Dataset (VSSD) by rendering 5 scenes using textured synthetic 3D models publicly available from Google 3D warehouse [26]. We have ensured to diversify scenes based on the following criterion:

- We have selected models that represent indoor, outdoor and object categories.
- The scenes are sufficiently large to capture large camera transformations.
- These scenes have non-homogeneous lightning conditions.
- Viewpoints in the scenes vary in texture.

The main motivation behind the effort is to provide a wide range of test cases for systematically evaluating visual servoing approaches on a common benchmark. All the scenes (CAD models) used in the dataset are publicly available and could be download at our project page[1].

1) Simulation results for 3D scene: In this experiment we aim to evaluate the control laws for our network architecture and to evaluate robustness in performing a positioning task. The initial pose (refer figure 6(a)) is selected from "House" model of VSSD dataset. The difference between desired and initial pose $\Delta r_{\text{desired}} = r_{\text{desired}} - r_{\text{init}} = [91.4\text{mm}, 92.3\text{mm}, 36.7\text{mm}, 8°, 10°, -5°]$. Although, the relative camera transformation is large, our approach is still able to converge to the desired pose with error in camera pose as $\Delta r_{\text{desired}} - \Delta r_{\text{final}} = [-5.1\text{mm}, 2.8\text{mm}, 0.5\text{mm}, -0.28°, -0.42° -0.42°,]$, which is around 4% in both translation as well as rotation. It could be seen from figure 6(e-g) that both the error as well as the camera velocity decrease exponentially despite the fact that these are output by a CNN. The experiment demonstrates that visuomotor representations are indeed learnt by our system. Also, the camera trajectory as shown in figure 6(h)is close to a straight line, which is desirable for visual servoing purposes.

2) Qualitative results on servoing dataset: The objective of this experiment is to show the efficacy of the proposed algorithm to servo to a diverse set of target instances across various environment and viewpoint variations. For every scene from the VSSD dataset, we evaluate our algorithm for two configurations of the initial and desired pose pair, with different transformations in 6 DOF. The resultant error images from figure 5 indicate that our CNN based approach is indeed able to attain the desired pose for large camera pose variations. Let us note that VSSD has non-homogeneous lighting conditions, hence the assumption of temporal luminance continuity made by previous featureless visual servoing approaches [3], [4] does not apply to such scenes. Also,

[1]http://robotics.iiit.ac.in/urls/d173716a.htm

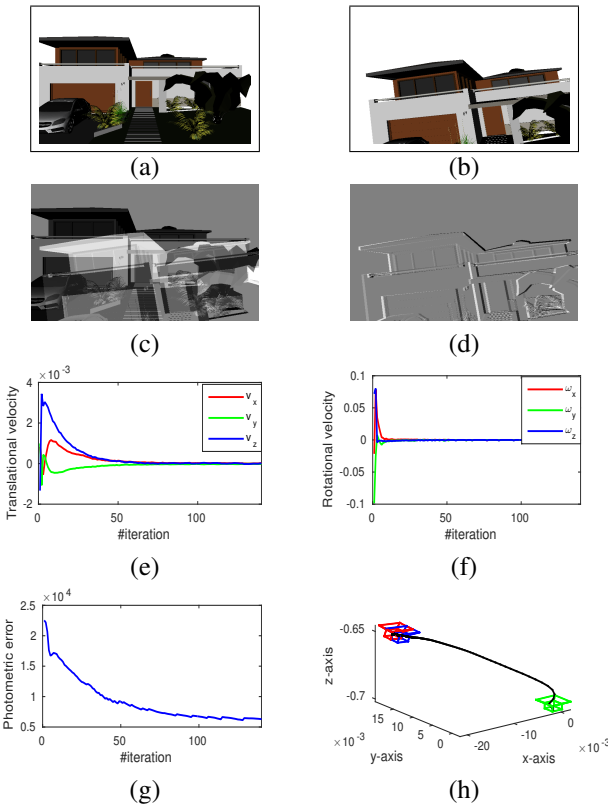


Fig. 6. **3D positioning task.** (a) Initial pose for a non-planar scene from house scene. (b) Desired pose. (c) Error image for initial image. Notice the large displacement of camera and variation in viewpoints. (d) Error image for resultant pose using CNN. (e) Translational velocity in m/s. (f) Rotational velocity in rad/s. (g) Photometric feature error. (h) Camera trajectory. Our approach is able to attain the desired pose even when the displacement between initial and desired pose is large and lightning is non-homogeneous.

the scene "kitchen" has textureless surfaces, which would make feature extraction difficult. This experiment validates the robustness of the feature representations learnt by the network for diverse and challenging environments without prior knowledge of the scene or camera used.

3) Real experiment using a quadrotor: In this experiment, we evaluate our approach on real world scenarios using a Parrot Bebop 2 drone. Since quadrotors are under-actuated, only 4 DOF tasks were selected for visual servoing. In real world, it is difficult to accurately predict the position of a drone. Hence we report the qualitative results and an approximate trajectory generated and reported by the drone by fusion of inertial measurement unit (IMU) , sonar sensor and optical flow sensor facing downward. Note that the images in the evaluation were not encountered during training of the CNN model. Again, the transformation between the initial and the desired pose is large. Precise convergence was not achieved since only 4 DOF could be controlled. Figure 7(a,b) show the initial and desired pose given to the CNN for generating velocity commands. The local controller aimed to track the quadrotor velocity commands generated by the CNN based high-level controller. The CNN forward pass processing was performed using a laptop computer with Core i7 CPU, Nvidia Quadro M2000M GPU and 16 GB RAM. It took 65ms for

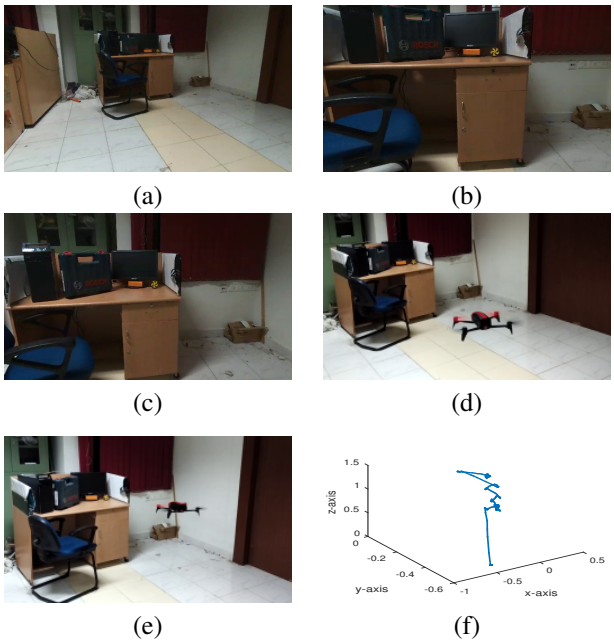


Fig. 7. **Positioning task using quadrotor.** (a) Initial pose for a real scene. (b) Desired pose. (c) Resultant pose at the end of approach. Notice the large displacement of camera and variation in viewpoints. (d) Initial position of quadrotor in the image space. (e) Final position of quadrotor in image space. (f) Approximate quadrotor trajectory in 3D.

one forward pass to complete on the machine. The drone was given 2 seconds to converge to the generated velocity before capture and forward pass of next image hence sending next velocity command. The image captured by the drone and corresponding control commands generated by the network were exchanged between the system and drone over WiFi channel.

VIII. Conclusion

In this work, we have introduced an end-to-end learning based framework for visual servoing tasks using CNN. The visuomotor representations learnt by the network generalises well across diverse environments. We have experimentally verified our approach on both synthetic as well as real world scenarios for robustness to non-homogeneous illumination and texture of scene. Unlike previous approaches, we do not need the knowledge of geometry of scene or camera parameters. Further, by learning the control representations we circumvent the requirement of any feature extraction or tracking step.

References

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

[2] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The confluence of vision and control*. Springer, 1998, pp. 66–78.

[3] C. Collewet and E. Marchand, "Photometric visual servoing," *IEEE TRO*, vol. 27, no. 4, pp. 828–834, 2011.

[4] E. Marchand and C. Collewet, "Using image gradient as a visual feature for visual servoing," in *IROS*, 2010, pp. 5687–5692.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[6] B. Glocker, S. Izadi, J. Shotton, and A. Criminisi, "Real-time rgb-d camera relocalization," in *ISMAR*, 2013, pp. 173–179.

[7] F. Chaumette and S. Hutchinson, "Visual servo control, part ii: Advanced approaches," *IEEE RAM*, vol. 14, no. 1, pp. 109–118, 2007.

[8] H. Pandya, K. M. Krishna, and C. Jawahar, "Servoing across object instances: Visual servoing for object category," in *ICRA*, 2015, pp. 6011–6018.

[9] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *ICRA*, 2016, pp. 512–519.

[10] T. Lampe and M. Riedmiller, "Acquiring visual servoing reaching and grasping skills using neural reinforcement learning," in *IJCNN*, 2013, pp. 1–8.

[11] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold, and C. Rother, "Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image," in *CVPR*, 2016.

[12] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *ICCV*, 2015, pp. 2938–2946.

[13] P. Agrawal, J. Carreira, and J. Malik, "Learning to see by moving," in *ICCV*, 2015, pp. 37–45.

[14] G. Costante, M. Mancini, P. Valigi, and T. A. Ciarfuglia, "Exploring representation learning with cnns for frame-to-frame ego-motion estimation," *RAL*, vol. 1, no. 1, pp. 18–25, 2016.

[15] K. Konda and R. Memisevic, "Learning visual odometry with a convolutional network," in *International Conference on Computer Vision Theory and Applications*, 2015.

[16] A. Dosovitskiy, P. Fischery, E. Ilg, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, T. Brox, *et al.*, "Flownet: Learning optical flow with convolutional networks," in *ICCV*, 2015, pp. 2758–2766.

[17] J. Campbell, R. Sukthankar, and I. Nourbakhsh, "Techniques for evaluating optical flow for visual odometry in extreme terrain," in *IROS*, vol. 4, 2004, pp. 3704–3711.

[18] A. Dosovitskiy, J. Tobias Springenberg, and T. Brox, "Learning to generate chairs with convolutional neural networks," in *CVPR*, 2015, pp. 1538–1546.

[19] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *ACMMM*, 2014, pp. 675–678.

[20] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *NIPS*, 2014, pp. 3320–3328.

[21] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks," *JMLR*, vol. 10, no. Jan, pp. 1–40, 2009.

[22] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," Robotics Institute, Tech. Rep., 2008.

[23] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *CVPR*, 2012, pp. 3354–3361.

[24] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The princeton shape benchmark," in *Shape Modeling Applications, 2004. Proceedings*, 2004, pp. 167–178.

[25] A. Handa, V. Ptrucean, S. Stent, and R. Cipolla, "Scenenet: An annotated model generator for indoor scene understanding," in *ICRA*, 2016, pp. 5737–5743.

[26] Google, "3d warehouse," 2014. [Online]. Available: http://sketchup.google.com/3dwarehouse/