2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Macau, China, November 4-8, 2019

Learning Real-time Closed Loop Robotic Reaching from Monocular Vision by Exploiting A Control Lyapunov Function Structure

Zheyu Zhuang[1], Jürgen Leitner[2], Robert Mahony[1]

***Abstract*— Visual reaching and grasping is a fundamental problem in robotics research. This paper proposes a novel approach based on deep learning a control Lyapunov function and its derivatives by encouraging a differential constraint in addition to vanilla regression that directly regresses independent joint control inputs. A key advantage of the proposed approach is that an estimate of the value of the control Lyapunov function is available in real-time that can be used to monitor the system performance and provide a level of assurance concerning progress towards the goal. The results we obtain demonstrate that the proposed approach is more robust and more reliable than vanilla regression.**

I. INTRODUCTION

Reaching, the process of moving a manipulator to a target pose defined relative to an observed environment, is a fundamental problem in robotics [1]. Performing reaching tasks in unstructured environments requires the system to explicitly or implicitly estimate the target pose of the manipulator using sensor input. Early work [2], [3] demonstrated the potential of vision as a sensor modality for reaching tasks and led to the field of visual servo control [4], [5]. One of the first approaches considered was to explicitly estimate target pose from vision input and then implement a classical pose controller to servo control the manipulator to the estimated pose [6]. Modern approaches that follow a similar philosophy use learning based pose estimation to identify grasp points [7], [8], [9], [10], [11] followed by classical control. Most recent work estimates target pose using RGBD sensors that provide depth information of the scene as well as the classical RGB images [12], [13], [14], [15]. A key challenge for such architectures is dealing with occlusion of the object by the manipulator end effector during grasping. Albeit that images including both end effector and object contain strong cues of relative pose important to the grasping problem.

Alternatively, control commands can be directly computed from images. Early works in this direction use hand-crafted image features and control the motion of the manipulator (camera) by linearising the image Jacobian [4]. This approach eliminates the necessity of using geometric object models to compute pose targets and reduces error in sensor modelling [6]. Modern algorithms that follow this philosophy use deep learning to learn the mapping from images to control (visuo-motor policy) end-to-end. Zhang *et al.* [16] achieve closed-loop reaching tasks by learning visuo-motor polices from monocular RGB images, although the mid-level regressor is pre-trained to estimate three Degrees of freedom target position. Levine *et al.* [17] learn a visuo-motor policy generated from trajectory-centric reinforcement learning algorithm with a CNN. One common issue with end-to-end visuo-motor policy learning is the regression targets for each joint control input are considered as approximating individual functions. This ignores the structural constraint that all joints are contributing to the end effector pose via manipulator's kinematic chain. Furthermore, such systems are unable to evaluate the quality of inferred control inputs. Levine *et al.* [18] and Viereck *et al.* [19] use CNNs to predict the quality (with respect to a grasping metric) of a collection of possible motion proposals and then choose the best action to achieve the goal at the cost of increasing inference time.

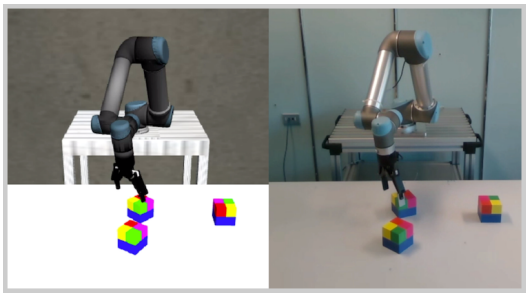

Fig. 1. We train a convolutional neural network to approximate a control Lyapunov function (cLf) to enable a robot to reach and grasp a coloured cube. We validated the performance of our approach in simulation and real-world experiments.

From a system and control theory perspective closed-loop reaching is a set-point regulation task. That is, one seeks a feedback law that makes the resulting autonomous closed-loop system globally exponentially stable with equilibrium at the desired pose. Any exponentially stable autonomous system admits a continuously differentiable positive definite Lyapunov function that decreases exponentially along solutions of the system [20]. A standard non-linear control design methodology [21], [22] is to first design a candidate control Lyapunov function (cLf) and then find a control law that ensures the exponential decrease of this function along closed-loop trajectories of the system. To the authors' knowledge, no prior work has been done on using visual sensors to directly learn control Lyapunov functions associated with a reaching task.

In this paper we propose a CNN architecture to learn the value of a control Lyapunov function for a multi-goal visual reaching task as well as the differentials of this

This research was supported by the Australian Research Council through the "Australian Centre of Excellence for Robotic Vision" CE140100016.

[1] Zheyu Zhuang, Robert Mahony are with "Australian Centre for Robotic Vision", Research School of Engineering, The Australian National University, Canberra ACT, 2601, Australia. first.last@anu.edu.au

[2] Jürgen Leitner is with the "Australian Centre for Robotic Vision", Queensland University of Technology (QUT), Brisbane, Australia.

function with respect to the control parameters (joint angles). The cLf differentials are then used as joint control inputs according to classical non-linear control design principles. Rather than just regress the six highly non-linear functions associated with the motion of each joint of the manipulator individually (vanilla regression). We propose a regularisation term that imposes a differential constraint between cLf and its derivatives during network training.

The goals of the paper are:

- Firstly, to demonstrate multi-goal visual grasping from a second person monocular camera feed using deep learning.
- Secondly, to show that learning a control Lyapunov function in concert with the control inputs results in additional reliability and performance compared to vanilla regression,
- Thirdly, to demonstrate learning the control Lyapunov function provides an estimate of the progress towards the goal during deep learned image based control and improves reliability of the closed-loop system.

By multi-goal reaching we consider the problem where there may be several equally 'good' possible reaching tasks that the robot may choose between. Rather than solve this problem in the classical 'decide then act' paradigm, where one particular goal is chosen and then a servo control for that goal is designed, we introduce a candidate control Lyapunov function which has a minimum at each of the possible 'good' reaching goals. By designing the control to minimise the cLf, the resulting behaviour is robust to changing environment conditions, removal of the goal, movement of goal, etc. An advantage of the proposed approach is that an estimate of the value of the cLf is available in real-time during the servo-control response. This value can be monitored and used as a performance measure of the system operation and provides a level of assurance concerning progress towards the goal. We use the on-line cLf value estimate as a proxy for convergence of the system and initiate a "hot swap" of the vision feed from a global image to a focused image (consisting of just the end effector and its local environment) when the cLf estimate descends below a pre-fixed threshold to aid the precision of the final stages of the servo-control task. The multi-goal capability and robustness to changing environments of the proposed approach is a key advantage that derives from the control Lyapunov function methodology. In addition, we demonstrate that by coupling the Lyapunov learning to the control input learning in a natural manner we significantly improve the reliability of the network compared to simply learning the control input using an end-to-end vanilla regression. We demonstrate the proposed performance through successful completion of a simple reaching task for grasping target cubes using only monocular vision from a second person point-of-view, a difficult geometric reconstruction task.

Section II presents the controller design. Section III describes the network architectures that we consider. In particular, we use a Siamese architecture to allow us to compute numerical derivatives of the network output in order to couple control inputs to the cLf intrinsically in the learning. The implementation details and learning results are included in Section IV. Section V documents an experimental study that evaluates the method's real-world performance against direct end-to-end learning and simulation.

II. Controller Design

This section presents the formulation of the control Lyapunov function for a reaching task. Note that the cLf definition can depend on any state of the manipulator or environment that is appropriate. For instance, Khansari-Zadehet al. learn a cLf for a reaching task based on human demonstration [23]. We formulate a control Lyapunov function based on six Degrees of Freedom object pose that is widely used in the literature.

A. Pose Representation

The six Degrees of Freedom (DoF) pose of the target and end effector are represented by elements of the Special Euclidean Group $\mathrm{SE}(3)$. Denote the pose of frame $\{A\}$ with respect to the reference frame $\{B\}$ as

$$ {}^{B}\boldsymbol{X}_{A} = \begin{bmatrix} {}^{B}\boldsymbol{R}_{A} & {}^{B}\boldsymbol{\xi}_{A} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathrm{SE}(3) $$

The left superscript is omitted if the pose defined with respect to the world reference frame. Denote the end effector frame as $\{H\}$ and target frame as $\{G\}$. The absolute end effector pose $\boldsymbol{X}_{H} = \boldsymbol{X}_{H}(\boldsymbol{\theta})$ is a function of joint angles $\boldsymbol{\theta} \in \mathbb{R}^{6\times 1}$, that is, the forward kinematics model of the manipulator. The relative pose of $\{G\}$ with respect to $\{H\}$ can be written

$$ {}^{H}\boldsymbol{X}_{G}(\boldsymbol{\theta}) = \boldsymbol{X}_{H}^{-1}(\boldsymbol{\theta})\, \boldsymbol{X}_{G} \tag{1} $$

B. Control Lyapunov Function Formulation

A cLf for a reaching task is a continuously differentiable scalar-valued positive-definite function $\mathcal{V}(\boldsymbol{\theta})$ of the joint angles that is zero only at the joint coordinates for the desired pose. Consider a collection of goals $\{G_j\}$ for $j = 1, \ldots, n$. The goal of a reaching task for a particular goal $\{G_j\}$ is to drive the target pose relative to end effector ${}^{H}\boldsymbol{X}_{G_j}(\boldsymbol{\theta})$ to the identity transformation. We define a control Lyapunov function for that goal to be

$$ \mathcal{V}_j := \| {}^{H}\boldsymbol{X}_{G_j} - \mathbf{1}_4 \|_F^2, \text{ for } j = 1, \ldots, n, $$

where $\mathbf{1}_4$ is the $\mathbb{R}^{4\times 4}$ identity matrix and $\| \cdots \|_F$ denotes the Frobenius norm. In order to deal with multiple goals we define a compound cLf

$$ \mathcal{V} := \min\{\mathcal{V}_1,\ \ldots,\ \mathcal{V}_n\}. \tag{2} $$

The multi-goal control Lyapunov function considered has a minimum (at zero) at each of the possible goals. Clearly, we have introduced saddle points in the cLf, where a descent criteria will separate system trajectory to either descend to one goal or another. Such a switching surface is inherently necessary in a multi-goal reaching task. However, we do not require the robot to choose which goal to target in advance,

and should the environment change during the reaching task, by a goal being moved or removed, the Lyapunov function will immediately conform to the new situation and the robot will continue to move towards the 'closest' available goal. This robustness to environment change is a highly desirable aspect of the proposed control architecture.

C. Velocity Controller Design

Let $\partial_{\boldsymbol{\theta}}\mathcal{V} = (\partial_{\boldsymbol{\theta}_1}\mathcal{V}, \ldots, \partial_{\boldsymbol{\theta}_6}\mathcal{V}) \in R^{1\times 6}$ denote the vector of partial differentials of the cLf (2). The velocity control proposed is

$$\dot{\boldsymbol{\theta}}_i := -\mu(t)\partial_{\boldsymbol{\theta}_i}\mathcal{V}, \text{ for } i = 1, \ldots, 6 \tag{3}$$

for a possibly time varying positive constant $\mu(t) > 0$. With this one has

$$\dot{\mathcal{V}} = \partial_{\boldsymbol{\theta}}\mathcal{V} \cdot \dot{\boldsymbol{\theta}} = -\mu(t)\|\partial_{\boldsymbol{\theta}}\mathcal{V}\|^2,$$

where $\dot{\boldsymbol{\theta}} = (\dot{\theta}_1, \ldots, \dot{\theta}_6)^\top$. The time-varying constant $\mu(t)$ is chosen to condition the control response for large Lyapunov values. This is particular important for a reaching task where the quadratic nature of the Lyapunov function chosen would lead to very aggressive control for large displacements. We choose

$$\mu(t) := \min\left\{\frac{1}{\mathcal{V}(\boldsymbol{\theta})}, 1\right\},$$

that is, for small value of $\mathcal{V}$ the control is not effected, but for large values of $\mathcal{V}$ the control is scaled down to be less aggressive.

To compute the individual $\partial_{\boldsymbol{\theta}}$ terms we use the velocity Jacobian $J(\boldsymbol{\theta})$ for the manipulator. That is, denoting the angular and translational rigid body velocity of $\boldsymbol{X}_H(\boldsymbol{\theta})$ expressed in the body-fixed frame by Ω and V respectively, one has $(\Omega, V)^\top = J(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}$. Let

$$\begin{pmatrix}\Omega\\ V\end{pmatrix}^\wedge = \begin{pmatrix} 0 & -\Omega_3 & \Omega_2 & V_1\\ \Omega_3 & 0 & -\Omega_1 & V_2\\ -\Omega_2 & \Omega_1 & 0 & V_3\\ 0 & 0 & 0 & 0\end{pmatrix} \in \mathfrak{se}(3)$$

where $\mathfrak{se}(3)$ is the matrix Lie-algebra of $\mathrm{SE}(3)$. Finally, we introduce an inner product on $\mathfrak{se}(3)$

$$\boldsymbol{M} = \begin{pmatrix}\mathbf{1}_3 & \mathbf{0}_3\\ \mathbf{0}_3 & \eta\mathbf{1}_3\end{pmatrix}, \tag{4}$$

for $\eta > 0$ a positive constant. The role of the inner product is to relatively weight variation in translation directions preferentially to rotations. Introducing a preferential weighting is necessary since the geometry, of the UR-5 robotic manipulator that we use, is such that relative small rotations of the end effector generate moderately large displacements of gripper position due to the offset of gripper tip from the wrist joint. This can be offset by making the control more aggressive in its translational components in the rigid-body velocity. In particular, a rigid-body velocity generated by a unit basis variation in i'th direction $\mathbf{e}_i$ is transformed into a variation $\boldsymbol{M}J(\theta)\mathbf{e}_i$. With these constructions, then

$$\partial_{\boldsymbol{\theta}_i}\mathcal{V}(\boldsymbol{\theta}) = 2\mathrm{tr}\left(({}^H\boldsymbol{X}_{G_j} - \mathbf{1}_4)^\top(\boldsymbol{M}J(\boldsymbol{\theta})\mathbf{e}_i)^\wedge\right), \quad \text{for } j = \arg\min\{\mathcal{V}_j(\boldsymbol{\theta})\}. \tag{5}$$

III. Learning the Control Lyapunov Function

Here we describe our approach to train a convolutional neural network to learn the control Lyapunov function (cLf) and its derivatives that will act as inputs for the closed-loop system from vision input. Let $\mathcal{D}$ denote the set of images from the vision sensor along with the robot joint angles.

$$\mathcal{D} = \{(\boldsymbol{I}, \boldsymbol{\theta}) : \boldsymbol{I} \in \mathbb{R}^{\text{width}\times\text{height}\times 3}, \boldsymbol{\theta} \in \mathbb{R}^{6\times 1}\}.$$

The goal of the network is to approximate the functions $\widehat{\mathcal{V}} : \mathcal{D} \to \mathbb{R}$, and $\widehat{\partial_{\boldsymbol{\theta}}\mathcal{V}} : \mathcal{D} \to \mathbb{R}^6$ using a regression network.

A. Network Architecture:

The proposed forward architecture for the network considered is shown in Fig. 2. The network can be split into three different parts, including a feature extractor, a joint angle decoder, and three fully connected layers.

Feature Extractor: The feature extractor is based on a relatively small (compared to modern day networks) CNN that was initially designed for image classification tasks, ResNet18 [24]. ResNet18 has 18 layers, of which one is a stand-alone convolutional layer, followed by eight convolutions using 2-layer residual blocks each and a final fully connected layer. For the proposed network architecture we remove the last FC layer and feed the feature neurons directly into the 'norm and concatenate' layer shown in Figure 2. We use the pretrained weights of the ResNet18 network to bootstrap.

The input image used is a monocular 224 $\times$ 224 RGB image taken from the second person point of view. The output feature used is the 512 value output from the ResNet18 architecture. Thus, one can write the output of the network as a function

$$f_\phi(\cdot) : \mathbb{R}^{224\times 224\times 3} \to \mathbb{R}^{512}$$

where ϕ represents the weights of a neural network in the feature extractor.

Joint Angle Decoder: Instead of using joint angles $\boldsymbol{\theta}$ as inputs to the network [16] or learning its interpretation with additional FC layers [25], [19]. we compute the forward kinematics explicitly and use them as a geometric joint feature descriptor. We do this by vectorising the six absolute joint poses $\{\boldsymbol{X}_1,\ \boldsymbol{X}_2,\ ...,\ \boldsymbol{X}_6\}$ to obtain an $\mathbb{R}^{72}$ vector that has the physical joint positions and orientation vectors as explicit elements:

$$g(\boldsymbol{\theta}) = \begin{bmatrix}\mathrm{vec}(\boldsymbol{R}_1) & \boldsymbol{\xi}_1^\top & \ldots & \mathrm{vec}(\boldsymbol{R}_6) & \boldsymbol{\xi}_6^\top\end{bmatrix}$$

where $\mathrm{vec}(\cdot)$ denotes the matrix vectorisation operand. Note that the camera coordinates of the i'th joint is then the projection (a linear operation) of three of the direct inputs to the network, while orientation information is similarly direct to compute in camera coordinates. The philosophy of this

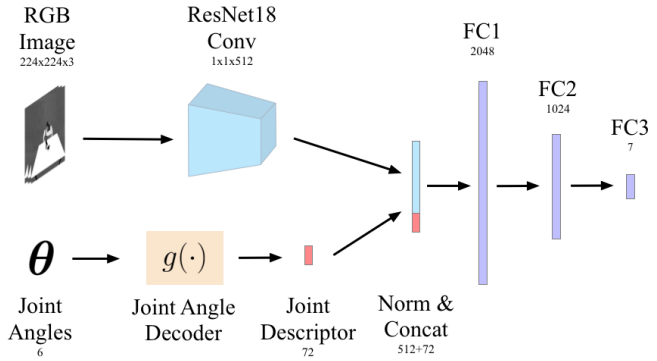

Fig. 2. Proposed regression network architecture. An RGB image $\boldsymbol{I}$ is fed into a feature extractor to generate a 512 dimensional descriptor. The joint angle decoder generates a 72 dimensional vector encoding the six joint absolute poses. The two descriptors are normalised, concatenated, and fed into the fully connected layers (FC). FC3 provides the 7 outputs consisting of the cLf $\widehat{\mathcal{V}}(\boldsymbol{\theta})$ and its 6 DoF partial derivatives $\widehat{\partial_{\boldsymbol{\theta}}\mathcal{V}}(\boldsymbol{\theta})$

approach is to save the network regressor from the need to approximate the highly non-linear forward kinematics of the manipulator.

Regressor: The three fully connected layers (using ReLU activiations), containing 2048, 1024, and 7 neurons respectively provides the regressor functionality for the network. The final 7 neurons are mapped to the cLf $\widehat{\mathcal{V}}(\boldsymbol{\theta})$ and its 6 DoF partial derivatives $\widehat{\partial_{\boldsymbol{\theta}}\mathcal{V}}(\boldsymbol{\theta})$.

The network regressor $h_\psi(\cdot)$ can be expressed as a function

$$h_\psi(\tilde{f}_\phi(\boldsymbol{I}), \tilde{g}(\boldsymbol{\theta})) \tag{6}$$

where ψ represents the weights of the FC part, and $\tilde{(\cdot)}$ represents the vector normalisation operand.

Note that there is no explicit requirement that the network outputs respect the differential constraint of the control Lyapunov function from which they are derived; that is, there is no explicit requirement that

$$\mathcal{V}(\boldsymbol{\theta} + \boldsymbol{\Delta\theta}) \approx \mathcal{V}(\boldsymbol{\theta}) + \partial_{\boldsymbol{\theta}}\mathcal{V}(\boldsymbol{\theta}) \circ \boldsymbol{\Delta\theta} \tag{7}$$

inherent in the network architecture. This dependency **must be learnt** by suitable choice of the loss function as discussed in the sequel.

B. Learning approach:

A naive, but straightforward, approach is to train the network using mean square error loss for both cLf and its six partial derivatives. Such a loss can be formulated, for a minibatch of m samples,

$$\ell_D = \frac{\alpha}{m}\sum_{k=1}^{m} \|\widehat{\mathcal{V}}_k - \mathcal{V}_k\|_F^2 + \frac{\beta}{6m}\sum_{k=1}^{m}\sum_{i=1}^{6} \|\widehat{\partial_{\theta_i}\mathcal{V}}_k - \partial_{\theta_i}\mathcal{V}_k\|_F^2 \tag{8}$$

where α and β are positive weighting factors. This loss treats each regressor as an individual objective and does nothing to explicitly encourage the differential constraint from Eq. (7) that couples the cLf value to its partial derivatives.

In essence the naive approach learns the given non-linear control $\widehat{\partial_{\theta_i}\mathcal{V}}$ from scratch in an end-to-end network architecture with vanilla regression method. In this case, learning the cLf value provides no additional robustness or reliability to the control, although it can be monitored during the closed-loop operation of the system to as a real-time performance measure.

C. Siamese Regressor

A key innovation of the proposed approach is the introduction of a Siamese regression architecture in order to **explicitly encourage** the differential constraint formulated in Eq. (7) in the regression.

For small variation in the joint angles we assume that the image obtained by the vision sensor will not vary significantly. Thus, we propose the Siamese architecture shown in Fig. 3 to use the same network weights ϕ for the feature extractor, along with different joint angle inputs to generate two separate values for the cLf. Let $\Delta\boldsymbol{\theta}_i \in \mathbb{R}^{6\times 1}$ be a small perturbation of the i'th joint angle. The output of the perturbed joint angle $\widehat{\mathcal{V}}(\boldsymbol{I}, \boldsymbol{\theta} + \boldsymbol{\Delta\theta_i})$ can be computed separately for each of the i joint angle perturbations and compared to the value $\widehat{\mathcal{V}}(\boldsymbol{I}, \boldsymbol{\theta})$ obtained for the unperturbed joint angle measurements. For a GPU implementation this sequential computation is achieved in a single forward pass by stacking seven decoded joint vectors (one original and six perturbed) along the batch dimension. Therefore it does not significantly impact the computational cost of the learning algorithm.

Based on this insight, we propose a loss term that encourages the differential constraint (7), for a mini-batch with m samples:

$$\ell_C = \frac{1}{6m}\sum_{k=1}^{m}\sum_{i=1}^{6} \left\| \widehat{\partial_{\theta_i}\mathcal{V}}_k(\boldsymbol{I}_k, \boldsymbol{\theta}_k) - \frac{\widehat{\mathcal{V}}(\boldsymbol{I}_k, \boldsymbol{\theta}_k + \Delta\boldsymbol{\theta}_i) - \widehat{\mathcal{V}}(\boldsymbol{I}_k, \boldsymbol{\theta}_k)}{\Delta\theta_i} \right\|_F^2. \tag{9}$$

Combining the direct regression loss (8) with (9) one obtains a combined loss

$$\mathcal{L} = \ell_D + \gamma\ell_C \tag{10}$$

where γ is a positive weight parameter.

The combined loss as in Eq. (10) is motivated by both the direct regression of the desired control, along with the requirement to apply the differential constraint from Eq. (7) explicitly in the network response to encourage the control input acts to decrease of the Lyapunov function.

D. Hot-Swap

During initial experiments, we found that the closed-loop response, although good for the macro motion of the manipulator, was unable to provide sufficiently precise positioning to achieve the desired reaching performance. Instead of increasing the resolution of the full input image of the network, we introduced an additional innovation in the control design, to train a "local network" that focuses on the near-end effector region. The philosophical concept is one of focused attention; that is, when we are close to

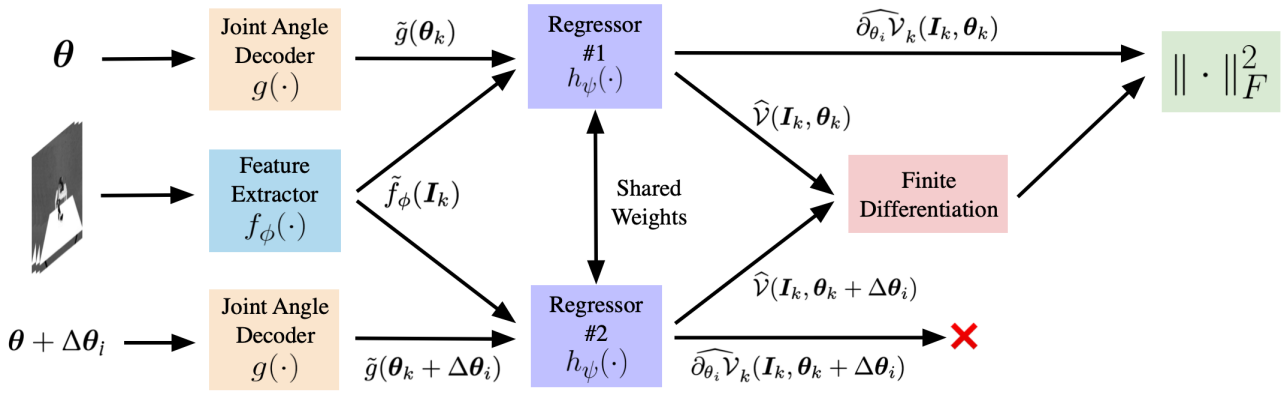

Fig. 3. Siamese regression for computing the i^{th} contrastive loss term given input pair $\{\boldsymbol{I}_k, \boldsymbol{\theta}_k\}$. Two identical regressors share the same weights and the normalised image feature $\tilde{f}_\phi(\boldsymbol{I}_k)$. Regressor #1 infers current cLf and gradient; regressor #2 computes corresponding values after perturbation by applying $\Delta\boldsymbol{\theta}_i$ to its secondary input. The original and perturbed cLf is used to approximate the partial derivative with respect to $\boldsymbol{\theta}_i$ numerical. The perturbed partial derivatives from second regressor is discarded. The i^{th} term of contrastive loss is computed as the squared L2 distance of the difference between inferred and numerically approximated partial derivative

a successful goal, the network can focus on only the local visual information. We do this by generating a local image input, separate from the global image, that is focused on the end effector of the robot (Fig. 4). This local image set is now used to train a second 'local' network that contains detailed visual information on the positioning task and is trained to regress *the same* outputs as the global network. The idea is to hot swap from the global network to the local network as the manipulator reaches the neighbourhood of the goal. Since both networks learn the same outputs the transition should be smooth and seamless, although in reality we know that numerical error will lead to some variation that will need to be smoothed at transition.

An additional advantage of the approach is that the control Lyapunov function value itself provides an ideal switching condition to initiate the hot swap. We hot-swap to the local network when the inferred cLf is lower than a designated threshold ζ. We introduce a small hysteresis, that the cLf value remains below an augmented threshold, $\mathcal{V} \leq (\zeta + \rho)$, for a small constant $\rho > 0$ to ensure smooth control handover. The convergence assurance provided by the cLF guarantees that the zoomed-in image will contain at least one target when control hand-over is triggered.

The local network uses an identical architecture to the global network but is trained only on zoomed-in images. Zooming to the local region relies on coarsely calibrated camera extrinsic parameters. We compute the projection of the 3D end effector pose (using the given joint positions and the forward kinematics) onto the image plane. A 130×130 region is cropped based on the projected coordinate from the original $1280 \times 768 \times 3$ raw camera image (see Fig. 4). The image is then upscaled to 224×224 to match the network input size. With global and local cLf regression network running in parallel, we are still able to achieve a 35Hz closed-loop control on average on a single GPU (GTX 1080 Ti).

IV. Implementation

To perform the reaching and grasping experiments, we use a Universal Robot UR5 6 DoF manipulator with Robotic Adaptive 2f-140 gripper. The vision sensor is a Chameleon3 RGB Camera with resolution set to 1280×768. The camera is approximately 1.6m high from the floor, 1.7m away from the manipulator base joint and inclined forward 21° towards the manipulator workspace (Fig. 5). Camera, manipulator and gripper communicate with the computer using ROS [26].

Collecting real-world manipulator data is a costly process [18]. Training a network with large-scale simulated dataset

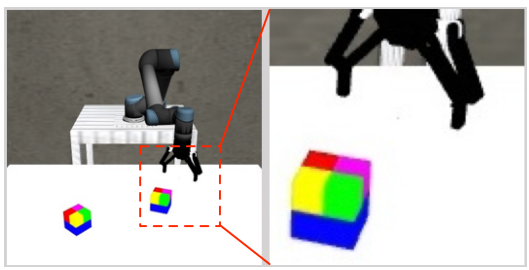

Fig. 4. The global and local network utilise different information from the same input image. The local network reduced the amount of information aby zooming onto the end effector region.

Fig. 5. Lab environmental set-up. The end effector shown is at the default home position. The camera's optical centre, table surface centre, and UR5 base joint are aligned.

is a viable alternative to collecting large sets of real world data [16], [19], [27]. We replicate the real-world set-up in Gazebo7 [28], as it is able to provide consistent ROS control between simulator and the real hardware. The simulator's configuration is geometrically identical (within measurement error) to that of the lab and differs only visually. The simulated camera is an ideal camera model with focal length set to that of the real camera's. Shadow rendering is turned off due to Gazebo7's inability in rendering realistic shadows under multiple light sources.

A. Dataset Collection

In this work, we use up to three identical targets to generate the multi-goal scenarios we consider. Each target has 70% probability of appearing in the workspace (1×0.4m on the table infront of the robot) for any given scenario. The location and rotation around Z axis for each target is sampled from uniform distributions.

The methodology used to sample the location of the manipulator in the training set makes a significant difference to the accuracy of the resulting network approximation. It is important to sample many times in the neighbourhood of the goal in order that the cLf and its derivatives are well modelled in the region in which the cLf is most structured, around its minimum. Further out from the goal the sampling can be much more coarse and achieve the same closed-loop performance. With a single target, we uniformly sample a fixed number of end effector positions on semi-spheres centred on the targets' centroid. For the global network, half of semi-spheres' radius are sampled from a zero mean normal distribution, the other half is sampled from uniform distribution between interval with lower bound set to zero. For the local network, the standard deviation of normal distribution and the upper bound of uniform distribution are reduced accordingly. At each end effector position, the Euler angles of the end effector frame are sampled on three separate uniform distributions with upper bounds constrained so that the end effector always pointing downwards. To handle multiple targets, we select one of the targets at a time and sample as above for that target, and repeat for the remaining. After collecting all end effector pose proposals, each full 6 DoF pose is verified by solving inverse kinematics.

B. Learning Details and Results

The dataset for training global cLf regression network has 140,000 simulated images, while the dataset for the local network has 35,000 samples. We train our networks with 90% of the training dataset and keep the remaining 10% for evaluation.

We apply random cropping, rotation, and colour jittering during training to provide for better real-world hardware calibration error tolerance. We randomly crop from the original image a 180×180 pixel patch and upsample it back to maintain the same input size to the network. The image is randomly rotated between -3° and 3° after random cropping. The brightness, saturation, contrast and hue of the input images randomly jitter at maximum 10%. The weights ϕ of the feature extractor are initialised from ResNet18 [24] pre-trained on ImageNet [29]. The network is implemented in Pytorch, and trained using Adam [30] optimiser for 80 epochs with batch size set to 64. The learning rate starts from 1×10^{-3} and is halved when the evaluation loss plateaus for 5 epochs.

Network Focus	Regress Method	Root Mean Square Error		Mean Relative Error	
		$\widehat{\mathcal{V}}(\boldsymbol{\theta})$	$\partial_{\boldsymbol{\theta}}\widehat{\mathcal{V}}(\boldsymbol{\theta})$	$\widehat{\mathcal{V}}(\boldsymbol{\theta})$	$\partial_{\boldsymbol{\theta}}\widehat{\mathcal{V}}(\boldsymbol{\theta})$
Global	**Siam.**	0.27	0.99	0.14	2.14
	Dir.	0.32	1.07	0.12	1.60
Local	**Siam.**	0.05	0.28	0.14	0.55
	Dir.	0.04	0.28	0.10	0.55

TABLE I

RESULTS FOR GLOBAL AND LOCAL NETWORKS SEPARATELY BEING TRAINED WITH BOTH SIAMESE AND DIRECT REGRESSION EVALUATED ON GLOBAL AND LOCAL DATASETS

In the Siamese loss ℓ_C (Eq. (9)), we set $\Delta\theta = 0.05$ for all six joint angles. The numerical differentiation method that is inherent in this approach inevitably introduces numerical error in the loss function that will tend to degrade the learning response. This is particularly the case when the Lyapunov function approaches a minimum and the 0.05 fixed displacement becomes large with respect to the variation in function value. In future work we intend to study whether more efficient methods to learn a differential constraint such as (7) are possible, however, for the moment we set the scaling factor γ in Eq. (10) is set to 0.5 for the global network and 0.1 for the local network. For both cases, the scaling factor α, β for cLf and gradient loss terms in in Eq. (9) are set to 1 and 6 respectively.

To compare our approach with vanilla regression, we separately train both global and local networks using direct regressing loss function ℓ_D from Eq. (8). The training is done on the identical training set, with identical parameters apart from the single parameter $\gamma = 0$ that 'turns off' the cost associated with (7).

Table I shows the learning evaluation results of the of four different networks: global and local networks trained with Siamese regression and direct regression loss functions. The training process for all four models stops after the evaluation loss plateaus. The results are shown in Table I. The Root Mean Square Error is the standard least squares error criteria, and is good at providing an overall indication of the quality of the estimation. The Mean Relative Error (MRE) is

$$\text{MRE} = \frac{\|\widehat{\mathcal{V}}_k(\boldsymbol{I}_k) - \mathcal{V}(\boldsymbol{\theta}_k)\|}{\|\mathcal{V}(\boldsymbol{\theta}_k)\|}$$

provides a better measure of the error relative to structure in the cLf and is important in evaluating the approximation close the target goal. We conclude from this statistical results firstly that both the global and local networks learn effectively, and secondly that there is no significant difference in the quality of learning with or without the proposed differential constraint. However, the second observation conflicts with the grasping experiments performed in both simulator

and real-world. This observation is further discussed in the next section.

V. Experiments and Results

Here we explain our experimental study that evaluates the method's simulated and real-world performance against vanilla regression method.

A. Siamese versus Direct Vanilla Regression

We perform two sets of real-world closed-loop servoing experiments to evaluate the performance of the proposed scheme, and to compare the Siamese and vanilla regression methods to evaluate the advantages of the Siamese network. The velocity control input is directly computed from the derivative outputs from the respective networks and Equation (3). The hot swap condition was taken from the value of the cLf.

Each set has 30 random samples – ten for one, two, and three targets respectively. For each sample, we test both Siamese and vanilla regression methods (with hot-swap mechanism) with only one grasp attempt. Two sets of experiments are performed under natural and artificial lighting conditions during daytime a night-time respectively.

A key observation of the implemented control is that the vanilla regression closed-loop system does not necessarily converge. There were numerous cases, 7 during simulation trials and 14 during real-world experiments (Table II) where the manipulator simply diverged from the goal under directly regressed closed-loop control. The most common failure mode was when the manipulator retracted towards the base, maintaining the end effector in the image, but pulling back until the workspace limits were reached. In contrast to this, the Siamese regression closed-loop response never diverged from the goal. We claim this is a direct consequence of the inherent robustness derived from enforcing the differential constraint (7).

A good benchmark for the six DoF reaching task is whether a grasp, made after convergence of the closed-loop reaching control, is successful. A successful grasp requires high precision positioning in both translation and rotation components of the gripper. Neither of the methods were infallible, even in simulation, under perfect conditions. We believe the baseline performance could be improved by using a more sophisticated network, however, the performance is sufficient to indicate the potential of the approach.

During daytime, Siamese and vanilla regression method achieved $80 \pm 7.3\%$ and $63.3 \pm 8.8\%$ grasp success rate respectively. Mean and standard deviation of the binomial distributions are computed using the standard minimum likelihood mean and variance estimators. During nighttime, the grasp success rate of directly regressed network degraded significantly, dropping to $26.7 \pm 8\%$. In comparison the Siamese regression maintained similar performance to the day level at $76.7 \pm 7.8\%$. The combined grasping results from two sets of experiments taken across a range of day and night conditions are given in Table II. Overall, the Siamese regression method achieves $78.3 \pm 5.4\%$ grasp success rate compared to the vanilla regression method at $45 \pm 6.4\%$. A two-sample t-test demonstrates greater than 99% confidence that the Siamese regression response outperforms the vanilla regression response.

From these results we conclude that design based on a control Lyapunov function and implementing a loss that encourages the differential constraint (7) leads to a significant improvement in grasping efficiency.

We also evaluated the reaching error in terms relative pose error ${}^{G}\boldsymbol{X}_{H}$. That is the mean accuracy of the final position achieved by the gripper, just prior to attempting to grasp (Table II). The displacement is measured in cm and the rotation error is measured in degrees in Euler angles. In order that these results are sensible, we excerpted the divergent cases from the vanilla regression response, as these would have adversely biased the comparison of precision. Comparing the statistical results from the real-world experiments in Table II, we find the performance difference, in terms of precision, between two regression methods are insignificant. The relative difference between average positioning error for the two methods is general several times smaller than the estimated error. Considering that both methods are effectively learning the same control law, this is not surprising.

Note that the learning evaluation results in Table I also shows no significant difference between two regression method during learning. We find this result interesting as it demonstrates that the measure of robustness for visuo-motor policy learning tasks is not easily visible in the statistical metrics that are commonly used in deep learning. It is also possible that the main advantage that could be gained by the differential constraint in precision is compensated adversely by the addition noise introduced in approximating the different of $\widehat{\mathcal{V}}(\boldsymbol{\theta})$ numerically.

B. Real-world Performance Degradation

In order to evaluate the performance difference between simulation and real-world, we repeat the same 60 trails in the simulation, including 20 for one, two, and three targets, using networks trained with both Siamese and vanilla regression methods in the simulation. The statistical results of these trials are shown in Table II. The statical results do not show significant differences between the real-world and simulation in terms of mean errors indicating that there is no bias between simulation and real-world response. However, the standard deviation in precision for the simulation is consistently smaller, especially associated with the yaw angle positioning. It is clear that the simulation response is significantly better, however, considering that no sim-to-real transfer learning techniques has been utilized we were encouraged by the relatively moderate depredation in grasp performance for the Siamese regression: grasp success rate drops from $95 \pm 2.28\%$ to $78.3 \pm 5.4\%$. In contrast, the network trained by direct regression starts from a lower base $75 \pm 5.6\%$, and drops by nearly half in performance $45 \pm 6.4\%$ when run on real data. Once again, we claim that the underlying robustness of the cLf approach has lead to the observed performance advantage.

		x (cm)	y (cm)	z (cm)	$roll$ (°)	$pitch$ (°)	yaw (°)	# Failed Grasps	# Diverge
Real	**Siam.**	$\mathbf{-0.6 \pm 4.7}$	$\mathbf{0.1 \pm 4.9}$	1.1 ± 2.4	6.3 ± 5.8	$\mathbf{-0.7 \pm 3.8}$	$\mathbf{-2.7 \pm 26.3}$	**13/60**	**0/60**
	Vanilla.	-0.7 ± 9.9	-2.2 ± 5.2	$\mathbf{-0.5 \pm 2.1}$	$\mathbf{3.6 \pm 5.4}$	-1.5 ± 4.1	-9.8 ± 17.5	27/46	14/60
Sim	**Siam.**	$\mathbf{-0.2 \pm 1.5}$	-1.3 ± 1.6	0.7 ± 1.3	7.8 ± 5.3	-0.8 ± 4.2	-1.7 ± 7.9	**3/60**	**0/60**
	Vanilla.	-0.3 ± 1.2	$\mathbf{-1.3 \pm 1.4}$	$\mathbf{0.01 \pm 1.06}$	$\mathbf{6.2 \pm 4.6}$	$\mathbf{-0.5 \pm 4.0}$	$\mathbf{-0.8 \pm 6.6}$	45/53	7/60

TABLE II

STATISTICAL RESULTS OF 60 GRASPING EXPERIMENTS (20 SAMPLES FOR ONE, TWO, AND THREE INSTANCES) IN REAL-WORLD AND SIMULATION USING BOTH SIAMESE AND VANILLA REGRESSION NETWORK WITH HOT-SWAP. THE POSE ERRORS ARE COMPUTED ONLY FOR CONVERGED CASES.

VI. CONCLUSIONS

This paper proposes a novel approach to learning control inputs from vision sensors for reaching tasks. The key contributions were: firstly, to demonstrate multi-goal visual grasping from a second person monocular camera feed using deep learning, secondly, to show that learning a control Lyapunov function in concert with the control inputs results in additional reliability and performance compared to vanilla regression, and thirdly learning a control Lyapunov function provides an estimate of the progress towards the goal during deep learned image based control.

REFERENCES

[1] C. C. Kemp, A. Edsinger, and E. Torres-Jara, "Challenges for robot manipulation in human environments [grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 20–29, 2007.

[2] Y. Shirai and H. Inoue, "Guiding a robot by visual feedback in assembling tasks." *Pattern recognition*, vol. 5, no. 2, pp. 99–106, 1973.

[3] G. J. Agin, "Computer vision systems for industrial inspection and assembly," *Computer*, no. 5, pp. 11–20, 1980.

[4] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE transactions on robotics and automation*, vol. 12, no. 5, pp. 651–670, 1996.

[5] P. I. Corke, "Visual control of robot manipulators–a review," in *Visual Servoing: Real-Time Control of Robot Manipulators Based on Visual Sensory Feedback*. World Scientific, 1993, pp. 1–31.

[6] P. I. Corke and S. A. Hutchinson, "Real-time vision, tracking and control," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 1. IEEE, 2000, pp. 622–629.

[7] A. Sahbani, S. El-Khoury, and P. Bidaud, "An overview of 3d object grasp synthesis algorithms," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 326–336, 2012.

[8] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesisa survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014.

[9] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1316–1322.

[10] S. Kumra and C. Kanan, "Robotic grasp detection using deep convolutional neural networks," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 769–776.

[11] D. Morrison, P. Corke, and J. Leitner, "Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach," *arXiv.org*, Apr. 2018.

[12] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," in *European conference on computer vision*. Springer, 2014, pp. 536–551.

[13] A. Krull, E. Brachmann, F. Michel, M. Ying Yang, S. Gumhold, and C. Rother, "Learning analysis-by-synthesis for 6d pose estimation in rgb-d images," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 954–962.

[14] A. Doumanoglou, V. Balntas, R. Kouskouridas, and T.-K. Kim, "Siamese regression networks with efficient mid-level feature extraction for 3d object pose estimation," *arXiv preprint arXiv:1607.02257*, 2016.

[15] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, and J. Xiao, "Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1386–1383.

[16] F. Zhang, J. Leitner, M. Milford, and P. Corke, "Sim-to-real transfer of visuo-motor policies for reaching in clutter: Domain randomization and adaptation with modular networks," *CoRR*, vol. abs/1709.05746, 2017. [Online]. Available: http://arxiv.org/abs/1709.05746

[17] 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.

[18] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, June 2017.

[19] U. Viereck, A. ten Pas, K. Saenko, and R. Platt, "Learning a visuomotor controller for real world robotic grasping using simulated depth images," *arXiv.org*, June 2017.

[20] H. K. Khalil, "Noninear systems," *Prentice-Hall, New Jersey*, vol. 2, no. 5, pp. 5–1, 1996.

[21] M. Krstic, I. Kanellakopoulos, and P. Kokotovic, *Nonlinear and Adaptive Control Design*. Providence, Rhode Island, U.S.A.: American Mathematical Society, 1995.

[22] R. Sepulchre, M. Janković, and P. Kokotović, *Constructive Nonlinear Control*. London: Springer-Verlag, 1997.

[23] S. Mohammad Khansari-Zadeh and A. Billard, "Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 752–765, June 2014.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," pp. 770–778, 2016.

[25] J. Leitner, A. W. Tow, N. Sunderhauf, J. E. Dean, J. W. Durham, M. Cooper, M. Eich, C. Lehnert, R. Mangels, C. McCool, P. T. Kujala, L. Nicholson, T. Pham, J. Sergeant, L. Wu, F. Zhang, B. Upcroft, and P. Corke, "The ACRV picking benchmark: A robotic shelf picking benchmark to foster reproducible research," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4705–4712.

[26] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[27] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," *arXiv preprint arXiv:1710.06537*, 2017.

[28] N. P. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator." in *IROS*, vol. 4. Citeseer, 2004, pp. 2149–2154.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*. IEEE, pp. 248–255.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.