Visual Servoing from Deep Neural Networks

Quentin Bateux[1], Eric Marchand[1], Jürgen Leitner[2], François Chaumette[3], Peter Corke[2]

***Abstract*—We present a deep neural network-based method to perform high-precision, robust and real-time 6 DOF visual servoing. The paper describes how to create a dataset simulating various perturbations (occlusions and lighting conditions) from a single real-world image of the scene. A convolutional neural network is fine-tuned using this dataset to estimate the relative pose between two images of the same scene. The output of the network is then employed in a visual servoing control scheme. The method converges robustly even in difficult real-world settings with strong lighting variations and occlusions. A positioning error of less than one millimeter is obtained in experiments with a 6 DOF robot.**

I. Introduction

Visual perception is important for humans and robots alike, it provides rich and detailed information about the environment the agent is moving in. The goal of visual servoing techniques is to control a dynamic system, such as a robot, by using the information provided by one or multiple cameras [1, 2]. Classical approaches to visual servoing rely on the extraction, tracking and matching of a set of visual features. These features, generally points, lines, or moments, are used as inputs to a control law that positions (or navigates) the robot in a desired pose. Many control strategies have been proposed over the years, in particular neural networks have been considered when designing control schemes early on [3, 4].

The tracking and matching of such features, especially given the rich and detailed information stemming from cameras, is a difficult task. While there has been progress in extracting the relevant features, a technique called direct visual servoing was introduced recently for exploiting the full image, requiring no feature extraction [5]. The main drawback of this direct approach is its small convergence domain compared to classical techniques. This is due to high non-linearities between the image information and the 3D motion. To remedy this issue we herein propose the use of a trained deep neural network to perform the extraction of features and estimation of the current image's pose relative to the desired. More precisely, the following contributions are described herein:

- re-purposing a commonly used deep neural network architecture, pre-trained for object classification, to perform relative camera pose estimation
- a novel training process, based on a single image (acquired at a reference pose), which includes the fast creation of a dataset using a simulator allowing for quick fine-tuning of the network for the considered scene. It also enables simulation of lighting variations and occlusions in order to ensure robustness.
- integrating the network with a position-based visual servoing control scheme robust to occlusions and variations in the lighting
- achieving precise positioning (sub-mm accuracy) on a 6 DOF robotic setup on planar scenes.

[1] Quentin Bateux and Eric Marchand are with Université de Rennes 1, IRISA, Inria, Rennes, France, Quentin.Bateux@irisa.fr, Eric.Marchand@irisa.fr

[2] Jürgen Leitner and Peter Corke are with the Australian Centre for Robotic Vision (ACRV), Queensland University of Technology (QUT), Brisbane, Australia j.leitner@roboticvision.org, peter.corke@qut.edu.au

[3] François Chaumette is with Inria, IRISA, Rennes, France Francois.Chaumette@inria.fr

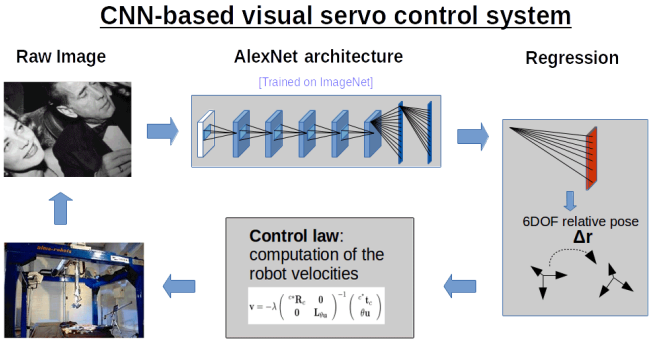


Figure 1. Overview of the proposed CNN-based visual servo control system

II. Related Work

Visual Servoing (VS) techniques have been applied to a variety of robotic tasks, including reaching, docking and navigation. While most of these approaches require a feature extractor or model tracker, direct VS was introduced to make use of the information available in full images [5]. The principle is to directly compare the current image with the desired image as a whole, avoiding the classical but error-prone feature detection, tracking and matching steps. Various control laws have been proposed in order to improve the robustness of direct VS approaches by varying descriptors or the cost functions, such as mutual information [6], histogram distances [7] or mixture of Gaussians [8].

The main drawback of these dense approaches is their small convergence domain, due to high non-linearities between the image information and the 3D motion. To remedy this issue, combining direct VS with other approaches has been proposed recently, such as the use of particle filtering in the control scheme [9] or to consider photometric moments to retain geometric information [10]. In this paper, we propose using deep learning and Convolutional Neural Networks (CNN) to direct VS schemes.

Over the last years deep neural networks, especially CNNs, have progressed the state-of-the-art in a number of computer vision tasks: image classification for object recognition [11], inferring a depth map from a single RGB image [12], computing displacement through homography estimation [13], and performing camera relocalization [14]. Deep learning has also started to become more prominent in robotics. For example, CNNs have also been trained for predicting grasp locations [15]. Recent progress in deep learning reaching tasks has achieved promising results: learning complex positioning tasks facilitated by vision [16, 17], coupled with reinforcement learning [18], and purely from vision without the use of any prior knowledge [19]. A hindrance to wide spread use is that these systems require large datasets and long training times. Our proposed approach partly overcomes this issue.

Due to the availability of easy to use frameworks, it is possible to construct, train and share deep neural networks and to build on networks already trained on very large datasets of millions of images. It is possible to re-purpose existing networks (with robust global descriptors already embedded in the lower layers) by using fine-

tuning techniques (such as in [20]). During tuning only the task-specific upper layers of the network are re-trained to perform a different task. In our case, rather than performing classification, the network is re-purposed to estimate the relative pose with respect to a desired image. The main advantage of using a network compared to the previous methods is that the deep learning approach can both create appropriate feature descriptors and also combine them in an optimized way for the designated task. This set of techniques is also well suited to real-time applications since once the training has been performed offline, the online computation of the task is fast (50ms on a middle-end graphics card), with little memory overhead, and most of all constant in term of computation costs, independently of the size and complexity of the dataset it was trained on.

III. CNN-BASED VISUAL SERVOING CONTROL SCHEME

The aim of a camera/end-effector positioning task is to reach a desired pose $\mathbf{r}^*$ starting from an arbitrary initial pose $\mathbf{r}$ (both $\in se(3)$). Image-based visual servoing (IBVS) is a method to control camera motion to minimize the positioning error between $\mathbf{r}$ and $\mathbf{r}^*$ in the image space [2].

A. From visual servoing to direct visual servoing

Considering the actual pose of the camera $\mathbf{r}$, the problem can therefore be written as an optimization process:

$$\widehat{\mathbf{r}} = \arg\min_{\mathbf{r}} \rho(\mathbf{r}, \mathbf{r}^*) \tag{1}$$

where $\widehat{\mathbf{r}}$ is the pose reached after the optimization (servoing) process, the closest possible to $\mathbf{r}^*$ if the system has converged (optimally $\widehat{\mathbf{r}} = \mathbf{r}^*$), and $\rho(.)$ is arbitrary cost function with a global minimum. For example, considering a set of geometrical features $\mathbf{s}$ extracted from the image, the goal is to minimize the error between $\mathbf{s}(\mathbf{r})$ and the desired configuration $\mathbf{s}^*$, which leads, by using as cost function the Euclidean norm of the difference between $\mathbf{s}$ and $\mathbf{s}^*$, to:

$$\widehat{\mathbf{r}} = \arg\min_{\mathbf{r}} \|\mathbf{s}(\mathbf{r}) - \mathbf{s}^*\|_{\mathbf{2}} \tag{2}$$

Visual servoing is classically achieved by iteratively applying a velocity command to the robot. This usually requires the knowledge of the interaction matrix $\mathbf{L_s}$ that links the temporal variation of visual features $\dot{\mathbf{s}}$ to the camera velocity $\mathbf{v}$:

$$\dot{\mathbf{s}}(\mathbf{r}) = \mathbf{L_s}\mathbf{v}. \tag{3}$$

This equation leads to the expression of the velocity that needs to be applied to the robot. The control law is classically given by [2]:

$$\mathbf{v} = -\lambda \mathbf{L_s^+}(\mathbf{s}(\mathbf{r}) - \mathbf{s}^*) \tag{4}$$

where λ is a positive scalar and $\mathbf{L_s^+}$ is the pseudo-inverse of the interaction matrix. To avoid the classical but error-prone extraction and tracking of geometrical features (such as points, lines, etc.), the notion of direct (or photometric) visual servoing has been introduced. It considers the image as a whole as the visual feature [5], ie. the set of features $\mathbf{s}$ becomes the image itself, $\mathbf{s}(\mathbf{r}) = \mathbf{I}(\mathbf{r})$. The optimization process can then be expressed as:

$$\widehat{\mathbf{r}} = \arg\min_{\mathbf{r}} \|\mathbf{I}(\mathbf{r}) - \mathbf{I}^*\|_{\mathbf{2}} \tag{5}$$

where $\mathbf{I}(\mathbf{r})$ and $\mathbf{I}^*$ are respectively the image seen at the pose $\mathbf{r}$ and the reference image (both composed of N pixels).

The main issue when dealing with direct visual servoing is that the interaction matrix $\mathbf{L_I}$ is ill-suited for optimization, mainly due to the heavily non-linear nature of the cost function, resulting in a small convergence domain. This is the same for the other features that have been considered in direct visual servoing (histogram distances, mutual information, etc.).

B. From direct visual servoing to CNN-based visual servoing

In this paper, we propose to replace the classical direct visual servoing [5], as described above, by a new scheme based on a convolutional neural network (CNN). The network is trained to estimate the relative pose between the current and reference image. Given an image input $\mathbf{I}(\mathbf{r})$ and the reference image $\mathbf{I}_0$, let the output of the network be:

$$\mathbf{\Delta^0 r} = net_{\mathbf{I}_0}(\mathbf{I}(\mathbf{r})) \tag{6}$$

with $\mathbf{\Delta^0 r} = ({}^{c_0}\mathbf{t}_c, \theta\mathbf{u})$ the vector representation of the homogeneous matrix ${}^{c_0}\mathbf{T}_c$ that expresses the current camera frame with respect to the camera frame associated to the reference image. (*Note:* $\theta\mathbf{u}$ is the angle/axis representation of the rotation matrix ${}^{c_0}\mathbf{R}_c$.)

If one wants to reach a pose related to a desired image $\mathbf{I}^*$, the CNN is first used to estimate the relative pose ${}^{c_0}\mathbf{T}_{c^*}$ (from $net_{\mathbf{I}_0}(\mathbf{I}^*)$), and then ${}^{c_0}\mathbf{T}_c$ (from $net_{\mathbf{I}_0}(\mathbf{I})$), from which $\mathbf{\Delta^* r}$ using ${}^{c*}\mathbf{T}_c = {}^{c_0}\mathbf{T}_{c^*}^{-1}\,{}^{c_0}\mathbf{T}_c$ is obtained.

Using the cost function $\rho(.)$ such as the Euclidean norm of the pose vector in Eq. (1), the minimization problem becomes

$$\widehat{\mathbf{r}} = \arg\min_{\mathbf{r}} \|\mathbf{\Delta^* r}\|_2 \tag{7}$$

which is known to present excellent properties [2]. Indeed, the corresponding control scheme belongs to pose-based visual servoing, which is globally asymptotically stable (ie. the system converges whatever the initial and desired poses are), provided the estimated displacement $\mathbf{\Delta^* r}$ is stable and correct enough. We recall that IBVS, and thus the schemes based on Eq. (2) and (7), can only be demonstrated as locally asymptotically stable for 6 DOF (ie. the system converges only if the initial pose lies in a close neighborhood of the desired pose). With our approach, the stability and convergence issues are thus moved from the control part to the displacement estimation part.

From $\mathbf{\Delta^* r}$ provided by the CNN, it is immediate to compute the camera velocity using a classical control law [2] :

$$\mathbf{v} = -\lambda \begin{pmatrix} {}^{c}\mathbf{R}_{c^*}\,{}^{c^*}\mathbf{t}_c \\ \theta\mathbf{u} \end{pmatrix} \tag{8}$$

By computing this velocity command at each iteration, it is then possible to servo the robot toward a desired pose solely from visual inputs.

C. Designing and training a CNN for visual servoing

In order to keep training time and the dataset size low, we present a method using a pre-trained network. Pre-training is a very efficient and widespread way of building on CNNs trained for a specific task. If a new task is similar enough, fine-tuning can be performed on the CNN so it may be employed in a different task. Since we work on natural images in a real-world robotic experiment, a pre-trained AlexNet [11] was chosen as a starting point. This network was trained on 1.2 million images from the ImageNet set, with the goal of performing object classification (1000 classes). While we are not interested in image classification, works such as [20] showed that it is possible to re-purpose a network by using the learned image descriptors embedded in the lower layers of an already trained AlexNet. This process, commonly referred to as fine-tuning, is based on the idea that certain parts of the network are useful to the new task and therefore can be transferred. Particularly the the lower layers (basic image feature extractors) will perform similar functionality

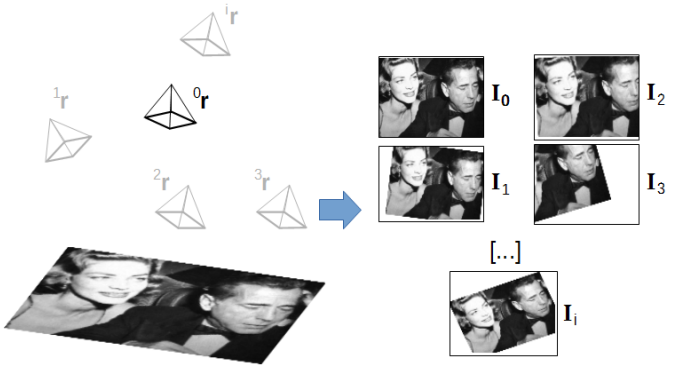


Figure 2. 3D plane with the projected reference image and multiple virtual cameras to generate multiple views of the scene.

in our relative pose estimation task. Only the upper layers require adaptation. Fine-tuning reduces training time (and data requirements).

We substitute the last layer – originally outputting 1000 floats with the highest representing the correct class – by a new layer that output 6 floats, ie. the 6 DOF pose. By replacing this layer, learned weights and connections are discarded and the new links are initialized randomly (see Figure 1). The resulting net is trained by presenting examples of the form ($\mathbf{I}$, $\mathbf{r}$), where $\mathbf{I}$ is a raw image, and $\mathbf{r}$ the corresponding relative pose as a training label. Since our training deals with distance between two pose vectors, we choose Euclidean cost function for network training, replacing the commonly used soft-max cost layer for classification, of the following form:

$$loss(\mathbf{I}) = ||\widehat{{}^{c_0}\mathbf{t}_c} - {}^{c_0}\mathbf{t}_c||_2 + \beta||\widehat{\theta\mathbf{u}} - \theta\mathbf{u}||_2 \tag{9}$$

$\widehat{{}^{c_0}\mathbf{t}_c}$ and $\widehat{\theta\mathbf{u}}$ respectively are the estimation of the translation and rotation displacements relatively to the reference pose. $\beta = 0.01$ is a scale factor to harmonize the amplitude of the translation (in m) and rotation (in degrees) displacements to facilitates the learning process convergence.

Starting from the trained AlexNet available for use with the Caffe library [21], the network was then fine-tuned. For this a new scene specific dataset of 11000 images with a variety of perturbations is created (as described in the next section). Using Caffe the network was trained with a batch size of 50 images over 50 training epochs.

The proposed method can be used with any kind of CNN network trained on images, therefore taking advantage of future developments in deep learned image classification. A thorough comparison of architectures is left for future work.

IV. Designing a Training Dataset

The design of the training dataset is the most critical step in the process of training a CNN, as it affects the ability of the training process to converge, as well as the precision and robustness of the end performances. As stated above we propose to fine-tune of a pre-trained network. Gathering real-world data is often cumbersome and sometimes unsuitable depending of the environment where the robot is expected to operate in. Furthermore, it can be difficult to re-create all possible conditions within the real-world environment. In this section we describe how simulated data allows us to generate a virtually unlimited amount of data. In addition we show how a variety of perturbations can be added which leads to satisfactory results without lengthy real-world data acquisition.

A. Creating the nominal dataset

The nominal dataset is the base of the training dataset. It contains all the necessary information for the CNN to learn how to regress from an image input to a 6 DOF relative pose. We will be adding various perturbations later on to ensure robustness when confronted with real-world data.

In our design, the nominal dataset is generated from a single real-world image $\mathbf{I_0}$ of the scene. This is possible by relying on simulation, in order to create images as viewed from virtual cameras. Figure 2 illustrate the image projected on a 3D plane and the varying (virtual) camera poses. This procedure generates datasets of thousand of images quickly (less than half an hour for 11k images) eliminating the time-consuming step of gathering real-world data. In comparison, 700 robot hours were necessary to gather 50k data points for a single task in [15].

The procedure to create the synthetic training dataset is then as follows (see also Figure 3):

- acquire a single image $\mathbf{I_0}$ at pose ${}^{\mathbf{0}}\mathbf{r}$, in order to get the camera characteristics and scene appearance
- create a 10,000 elements dataset, consisting of tuples $({}^{i}\mathbf{r}, \mathbf{I}_i)$, through virtual camera views in simulation (as illustrated in Figure 2). The first 10,000 virtual camera poses are obtained using a Gaussian draw around the reference pose ${}^{0}\mathbf{r}$, in order to have an appropriate sampling of the parameters space (the 6 DOF pose). The scene in the simulator is set up so that the camera-plane depth at ${}^{0}\mathbf{r}$ is equivalent to 20cm, and the variances for the 6DOF Gaussian draw are such as (1cm, 1cm, 1cm, 10°, 10°, 20°), respectively for the $(\mathrm{t}_x, \mathrm{t}_y, \mathrm{t}_z, \mathrm{r}_x, \mathrm{r}_y, \mathrm{r}_z)$ DOF.
- the dataset is appended with 1,000 more elements. These are created by a second Gaussian draw with smaller variances (1/100 of the first draw). The finer sampling around ${}^{\mathbf{0}}\mathbf{r}$ enables the sub-millimeter precision at the end of the robot motion.

B. Adding perturbations to the dataset images

In order to obtain a more robust process, two main perturbations were modeled and integrated in the dataset, namely, lighting changes (both global and local) and occlusions. We assume the scene to be static under nominal conditions for each experiment (ie. no deformations or temporal changes in the structure).

1) Modeling illumination variations with 2D Gaussian functions: Lighting conditions are a common problem when dealing with real-world images. These are of global and local nature. In order to model the global light, one can simply alter the pixel intensities by considering affine variation of the intensities. Local lighting changes are more challenging and to obtain realistic synthetic images time-consuming rendering algorithms are required. We alleviate this issue by working with planes in 3D space only, allowing to model lights as local 2D light sources and get realistic results. For each image chosen to be altered, the following mixture of 2D Gaussians is applied at each pixel (x, y):

$$\mathbf{I}_l(x,y) = \sum_{l=1}^{N_{lights}} \mathbf{I}(x,y) f_l(x,y) \tag{10}$$

Each 2D Gaussian in turn can be modelled as

$$f_l(x,y) = Ae^{-\left(\frac{(x-x_0)}{2\sigma_x^2} + \frac{(y-y_0)}{2\sigma_y^2}\right)} \tag{11}$$

where (x_0, y_0) (in pixel units) corresponds to the projection of the center of the simulated directional light source, gain A to its intensity,

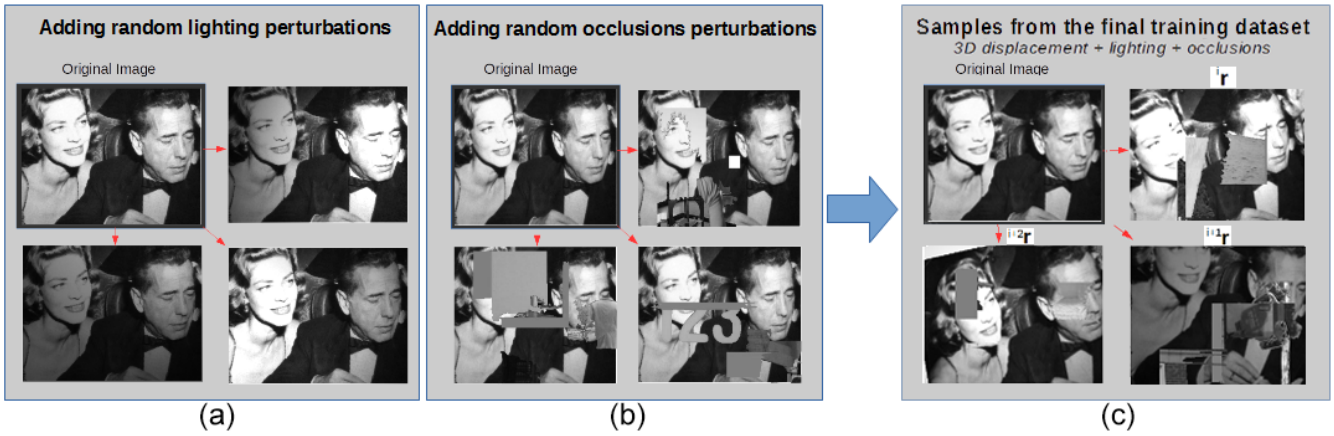


Figure 3. Overall process to create a training set from the original input image: (a) examples after applying local illumination changes; (b) examples after adding super-pixels from random images as occlusions; (c) examples from the final dataset after applying all perturbations.

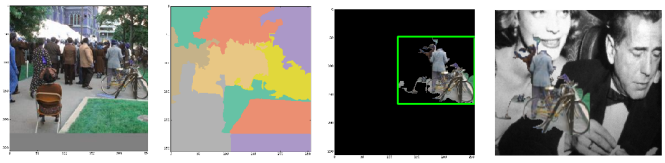


Figure 4. Synthetic occlusion generation: on an arbitrary images in the *Label-Me* dataset (left) segmentation is performed. A segmented cluster is selected at random. It provides a coherent "occlusion" patch, which is merged with a dataset image and added to the dataset (last image).

and (σ_x, σ_y) to the spread along each axis. An example of the resulting images can be seen in Figure 3(a). We purposely let out the modeling of specularities, as the material and reflection properties are unknown. Our method will handle them as a sub-class of occlusions (see next section).

2) Modeling occlusions by superimposing coherent pixel clusters from other image datasets: Dealing with occlusions is challenging due to the potential variety in size, shape and appearance that can appear in a real environment. Additionally, when training a CNN, one has to be careful to create the training set with a variety of perturbations included. This is to prevent the network from over-fitting on the presented examples and thus being unable to handle real world occlusions. We present here a procedure to provide the network with a realistic set of occlusions with an adequate range in size, shape and appearance.

To address this issue, we are adding clusters of pixels – representing a coherent part of an image – from other datasets and superimpose them on the previously generated images. To create somewhat realistic conditions real world images were preferred over synthetic occlusion images. These images provide a variety of scenes that represent interesting and varied occlusions, rather than those generated from geometrical or statistical methods. Herein the *Label-Me* dataset [22] containing thousands of outdoor scene images was chosen. The scenes contain a variety of objects in a wide range of lighting conditions. We then applied the following work-flow (illustrated in Figure 4) to each of the images in our simulated dataset that we want to alter:

- select randomly one image from the *Label-Me* dataset;
- perform a rough segmentation of the image by applying the SLIC super-pixel [23] segmentation algorithm creating coherent pixel groups (implementation available in OpenCV)
- we select a random cluster from the previous step, and then insert this cluster into the image to alter at a random position.

This method allows us to get a training dataset with randomly varied occlusions such as illustrated in Figure 3(b). By stacking the two described perturbations on our initial nominal dataset, we are able to generate a final training dataset with all the desired characteristics, as shown in Figure 3(c).

V. Experimental Results on a 6 DOF Robot

This section describes a set of experiments performed on an Afma 6 DOF gantry robot in a typical eye-in-hand configuration. At the beginning of each experiment, the robot is moved to an arbitrary starting pose $\mathbf{r_0}$ and the task is to navigate the robot back to a position defined by a desired image.

A. Nominal Conditions

In terms of pose offset, the robot has to perform a displacement given by $\mathbf{\Delta^0 r} = ({}^{c_0}\mathbf{t}_c, \theta\mathbf{u})$:

$$
\begin{aligned}
{}^{c_0}\mathbf{t}_c &= (1\text{cm}, -24\text{cm}, -9\text{cm}), \\
\theta\mathbf{u} &= (-10^\circ, -16^\circ, -43^\circ),
\end{aligned}
$$

with a distance between the camera and the planar scene of 80cm at the desired pose $\mathbf{r}^*$. Figure 5(f) shows the image at the final pose. Figure 5(h) shows the image error between the final and desired image. The training of the network with the 11000 images was performed offline. Figures 5(a) through 5(d) show that our CNN-based direct visual servoing approach converges without any noisy nor oscillatory behaviours when performed in a real-world robotic setting. Furthermore, the position of the system at the end of the motion is less than one millimeter from the desired one. No particular efforts were made to have "perfect" lighting conditions, but also no external lighting variations or occlusions were added. These were introduced in the next experiment.

B. Dealing with perturbations: Light changes and occlusions

Given the same initial conditions as above additional light sources and external occlusions were added to test the robustness of our approach. The robot captures a single image at the initial pose, the network is trained again and then our CNN-based direct visual servoing is performed. While the robot is servoing the light coming from 3 lamps is changed independently, resulting in global and local

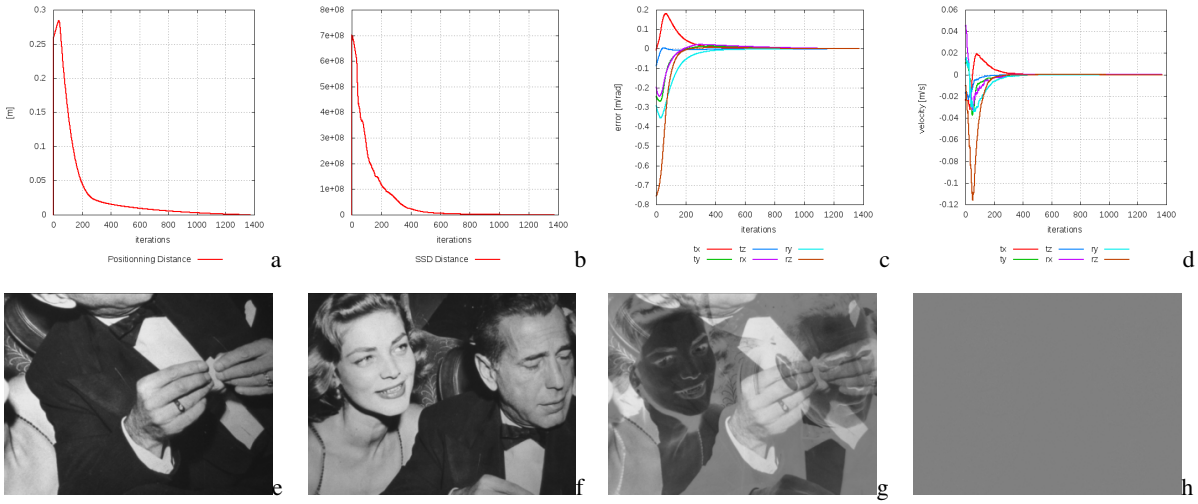


Figure 5. CNN-based visual servoing on a planar scene; (a) Positioning error; (b) SSD distance; (c) Translational and rotational errors; (d) Camera velocities in m/s and rad/s; (e) Image at initial pose $\mathbf{I_0}$; (f) Image at final pose $\mathbf{I}(\widehat{\mathbf{r}})$; (g) Image error $\mathbf{I_0} - \mathbf{I}^*$ at initial pose; (h) Image error $\mathbf{I}(\widehat{\mathbf{r}}) - \mathbf{I}^*$ at the final pose

light changes. In addition during the experiment various objects are added, moved and removed from the scene in order to create occlusions.

Figure 6 shows the graphs plotted for this second experiment.We can see that despite the variety and severity of the applied perturbations, the control scheme does not diverge. The method instead exhibits only a loss in final precision, ranging from 10cm (in accumulated translation errors) at the worst of the perturbations (samples are depicted in Figure 7 and show the various perturbations that occurred) to less than one millimeter error when back in the nominal conditions. A slower convergence is also observed when compared with the nominal conditions experiment. Additionally, most of the positioning error lies in the coupled translation and rotation degrees-of-freedom tx/ry and ty/rx. This keeps most of the scene in the camera's field of view by keeping the center of the scene aligned with the optical axis. The robot can therefore effectively reach the desired pose once the perturbed conditions are removed. This resilience is highlighted at iterations 100 and 260 when very strong perturbations occur as the operator's hand briefly occlude most of the scene, inducing a strong spike in both the SSD and the velocities applied to the robot (as the network receives as input only non-relevant information). However, as soon as this perturbation vanishes, the method is able to retrieve instantaneously its converging motion. It is important to note that since no tracking is involved, no elaborate scheme were introduced to deal with sudden loss of information and re-initialization of the method as it is able to regains its performances as soon as the information becomes available again. It also can be seen that the perturbations observed on the outputs of the network (Figure 6(c)), which are used as inputs of the control scheme, are not synchronous and are less noisy than the perturbations observed in the sum-of-squared-distances (SSD) plot (Figure 6(b)).

VI. Conclusion and Perspectives

In this paper we presented a new generic method for robust visual servoing from deep neural networks. We re-purpose pre-trained convolutional neural network (CNN) by substituting the last layer with a new output layer. Together with a matched general cost function, it enables enable fine-tuning of CNNs for visual servoing tasks.

Using a regression layer rather than a classification one as output layer re-configures the neural network to estimate the the relative pose to the desired image at each frame. Selection of the right dataset is critical for training a neural network, and we herein present an approach to design and collect a synthetic dataset for quick fine-tuning of the network to facilitate visual servoing. The synthetic data includes multiple views, local illumination changes from simulated 3D light sources, and simulated occlusions using coherent patches from randomly selected real-world image datasets.

We demonstrated the validity and efficiency of this approach with experiments on a 6 DOF gantry robot. The proposed method achieves millimeter accuracy through all 6 degrees of freedom in centimeter- and meter-scale positioning tasks. Furthermore we have demonstrated that the proposed approach is robust to strong perturbations as lighting variations and occlusions. The current framework allows a robot to visual servo with respect to a single scene, which forms the basis of the training set. Changing the application scene only requires synthesis of a new, comparatively small, training dataset and fine-tuning of the network to generate the desired pose estimates. Future research will focus on extending the proposed method to generalize to multiple scenes (including 3D ones), eventually training a network that provides scene-agnostic relative camera pose estimations.

References

[1] S. Hutchinson, G. Hager, and P. Corke. "A tutorial on Visual Servo Control". In: *IEEE Trans. on Robotics and Automation* 12.5 (Oct. 1996), pp. 651–670.

[2] F. Chaumette and S. Hutchinson. "Visual Servo Control, Part I: Basic Approaches". In: *IEEE Robotics and Automation Magazine* 13.4 (Dec. 2006), pp. 82–90.

[3] B. Kroze, M van der Korst, and F. Groen. "Learning strategies for a vision based neural controller for a robot arm". In: *IEEE Workshop on Intelligent Motion Control*. 1990, pp. 199–203.

[4] G. Wells, C. Venaille, and C. Torras. "Promising research vision-based robot positioning using neural networks". In: *Image and Vision Computing* 14.10 (1996), pp. 715–732.

[5] C. Collewet and E. Marchand. "Photometric visual servoing". In: *IEEE Trans. on Robotics* 27.4 (Aug. 2011), pp. 828–834.

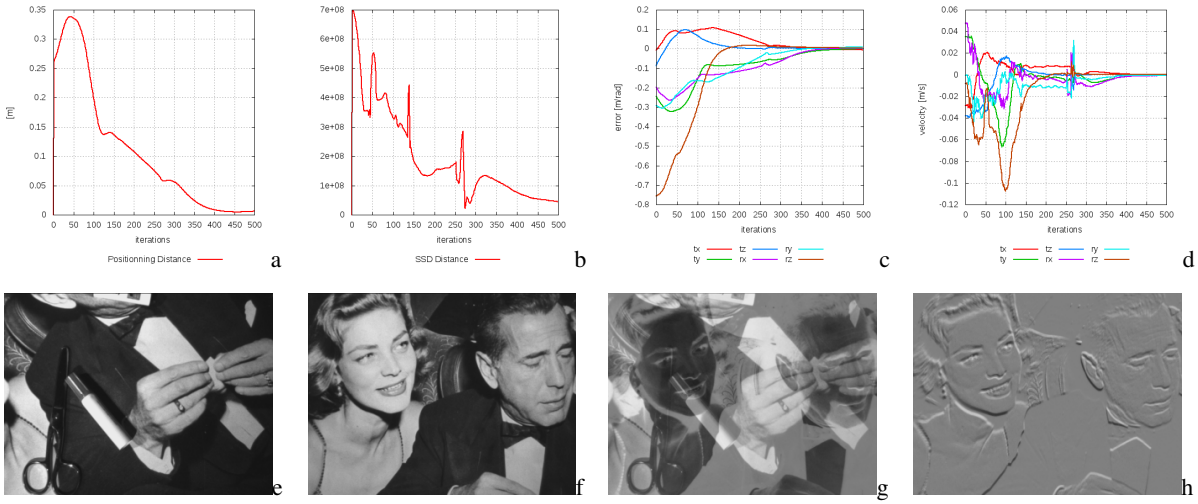


Figure 6. CNN-based visual servoing on a planar scene with various perturbations. (a) Positioning error; (b) SSD distance; (c) Translational and rotational errors; (d) Camera velocities in m/s and rad/s; (e) Image at initial pose $\mathbf{I_0}$; (f) Image at final pose $\mathbf{I}(\widehat{\mathbf{r}})$; (g) Image error $\mathbf{I_0} - \mathbf{I}^*$ at initial pose; (h) Image error $\mathbf{I}(\widehat{\mathbf{r}}) - \mathbf{I}^*$ at the final pose

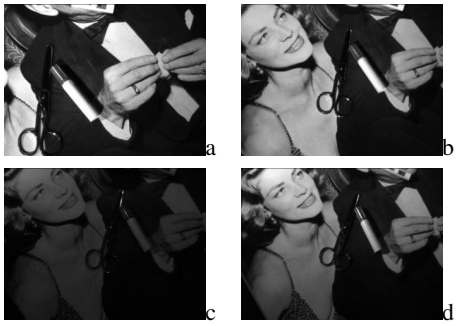


Figure 7. Images collected during real-world experiments on our 6DOF robot. Significant occlusions and variation in the lighting conditions can be seen.

[6] A. Dame and E. Marchand. "Mutual information-based visual servoing". In: *IEEE Trans. on Robotics* 27.5 (Oct. 2011), pp. 958–969.

[7] Q. Bateux and E. Marchand. "Histograms-based Visual Servoing". In: *IEEE Robotics and Automation Letters* 2.1 (Jan. 2017), pp. 80–87.

[8] N. Crombez, G. Caron, and E. M. Mouaddib. "Photometric Gaussian mixtures based visual servoing". In: *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2015, pp. 5486–5491.

[9] Q. Bateux and E. Marchand. "Particle filter-based direct visual servoing". In: *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2016, pp. 4180–4186.

[10] M. Bakthavatchalam, F. Chaumette, and E. Marchand. "Photometric moments: New promising candidates for visual servoing". In: *IEEE Int. Conf. on Robotics and Automation (ICRA)*. 2013, pp. 5521–5526.

[11] A. Krizhevsky, I. Sutskever, and G. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[12] D. Eigen, C. Puhrsch, and R. Fergus. "Depth map prediction from a single image using a multi-scale deep network". In: *Advances in neural information processing systems*. 2014, pp. 2366–2374.

[13] D. DeTone, T. Malisiewicz, and A. Rabinovich. "Deep image homography estimation". In: *Robotics: Science and Systems 2016, Workshop on Limits and Potentials of Deep Learning in Robotics*. arXiv preprint arXiv:1606.03798. 2016.

[14] A. Kendall, M. Grimes, and R. Cipolla. "PoseNet: A convolutional network for real-time 6-DOF camera relocalization". In: *IEEE Int. Conf. on Computer Vision, CVPR'2015*. 2015, pp. 2938–2946.

[15] L. Pinto and A. Gupta. "Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours". In: *IEEE Int. Conf. on Robotics and Automation (ICRA)*. Stockholm, Sweden, May 2016, pp. 3406–3413.

[16] S. Levine et al. "End-to-end training of deep visuomotor policies". In: *Journal of Machine Learning Research* 17.39 (2016), pp. 1–40.

[17] C. Finn et al. "Deep spatial autoencoders for visuomotor learning". In: *IEEE Int. Conf. on Robotics and Automation (ICRA)*. Stockholm, Sweden, 2016, pp. 512–519.

[18] S. Lange, M. Riedmiller, and A. Voigtlander. "Autonomous reinforcement learning on raw visual input data in a real world application". In: *Int. Joint Conf. on Neural Networks*. June 2012, pp. 1–8.

[19] F. Zhang et al. "Towards vision-based deep reinforcement learning for robotic motion control". In: *Australasian Conf. on Robotics and Automation (ACRA)*. 2015.

[20] S. Karayev et al. "Recognizing image style". In: *British Machine Vision Conference (BMVC)*. 2013.

[21] Y. Jia et al. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *ACM Int. Conf. on Multimedia, MM'14*. Orlando, FL, 2014, pp. 675–678.

[22] B. Russell et al. "LabelMe: a database and web-based tool for image annotation". In: *Int. Journal of Computer Vision* 77.1-3 (2008), pp. 157–173.

[23] R Achanta et al. "SLIC superpixels compared to state-of-the-art superpixel methods". In: *IEEE transactions on pattern analysis and machine intelligence* 34.11 (2012), pp. 2274–2282.