Training Deep Neural Networks for Visual Servoing

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 positioning tasks by visual servoing. A convolutional neural network is fine-tuned to estimate the relative pose between the current and desired images and a pose-based visual servoing control law is considered to reach the desired pose. The paper describes how to efficiently and automatically create a dataset used to train the network. We show that this enables the robust handling of various perturbations (occlusions and lighting variations). We then propose the training of a scene-agnostic network by feeding in both the desired and current images into a deep network. The method is validated on a 6 DOF robot.**

I. Introduction

The goal of visual servoing (VS) techniques is to control a dynamic system, such as a robot, by using the data provided by one or multiple cameras [13], [6]. Classical approaches 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 allows a robust positioning of the robot. While there has been progress in extracting and tracking the relevant features, a new approach called direct visual servoing (DVS, eg, [10], [7], [16]) was introduced recently to consider the image as a whole, which does not require anymore feature extraction nor tracking. The main drawback of DVS is its small convergence domain compared to classical techniques, which is due to the high non-linearities of the cost function to be minimized.

Various schemes have been proposed in order to improve the robustness of DVS by considering various descriptors (image intensity, gradient, color, etc.) or cost functions (mutual information [9], histogram distances [4], mixture of Gaussians [8]). To increase the convergence domain, combining DVS with other approaches has been proposed recently, such as using photometric moments to retain geometric information [2] or particle filtering in the control scheme [3]. To alleviate these issues we herein propose to bypass the modeling step thanks to a trained deep neural network.

Although learning techniques (eg, [14]) and classical neural networks [20], [29] have been occasionally investigated in VS, machine learning did not receive much interest in the past. Over the last years, deep neural networks, especially Convolutional Neural Networks (CNN), have improved the state-of-the-art in a number of computer vision tasks: image classification and object recognition [19], depth map inference from a single image [12], motion estimation [11], pose estimation [25], camera relocalization [18]. This diversification of applications of CNNs was fueled in part by the possibility to build on already-trained networks through common deep learning libraries such as Caffe [15] (used to implement the proposed method), and by using fine tuning techniques [17]. Meanwhile, deep learning has also started to become more prominent in robotics. For example, CNNs have been trained for predicting grasp locations [24], and for learning to achieve complex positioning tasks in joint space from raw pixel data by applying reinforcement learning techniques [22], [30]. Recently, there has been a renewed interest in applying CNNs for VS, such as [21] that demonstrated that it is possible to use features embedded in a deep neural network for learning predictive dynamic models. This work has been validated in simulation for a 4 DOF visual target following task. In [27], the authors train a CNN to perform 6 DOF end-to-end VS positioning task by training a network on an available navigation dataset.

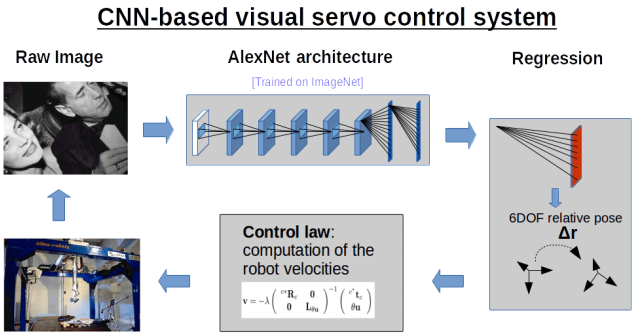

Fig. 1. Overview of the proposed CNN-based VS control system

In this paper, we propose to generate synthetic datasets on-demand, allowing for more control on the dataset characteristics (perturbations, number and variability of examples...), fitting it closely to the target task. A method is proposed to perform eye-in-hand VS from a CNN that estimates the relative pose between two images. This relative pose is then fed into a classical control scheme. More precisely, the following contributions are described herein:

- a pre-trained CNN is re-purposed to perform relative pose estimation. We also show that the proposed approach is efficient considering two different pre-trained networks with different architectures: Alexnet [19] and

A preliminary version of this paper has been presented as a poster in RSS Wk on New Frontiers for Deep Learning in Robotics and described in Arxiv preprint [5]. [1] Quentin Bateux and Eric Marchand are with Univ Rennes, Inria, CNRS, IRISA, 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, Univ Rennes, CNRS, IRISA, France Francois.Chaumette@inria.fr

VGG [28]. Thus it could benefit from improvement in future network architecture.
- an extension of the approach toward a scene-agnostic scheme is also proposed.
- a novel training process is introduced, 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 to ensure robustness within the learning phase.
- precise positioning is achieved (sub-millimeter accuracy) on a 6 DOF robotic setup on both planar and 3D scenes.

II. CNN-BASED VS CONTROL SCHEME

A. From VS to DVS

Any VS can always be written as an optimization problem [23]. The goal of VS is that, from an initial arbitrary pose, the robot reaches the desired pose $\mathbf{r}^*$ that best satisfies some properties measured in or from the images. If we note ρ, the function that measures the positioning error, then the VS task can be written as:

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

where $\widehat{\mathbf{r}}$ is the pose reached at the end. VS can therefore be considered as an optimization of the function ρ where $\mathbf{r}$ is incrementally updated to reach an optimum of ρ at $\widehat{\mathbf{r}}$. If ρ is correctly chosen, the final pose $\widehat{\mathbf{r}}$ at the end of the minimization should be equal to the desired one $\mathbf{r}^*$.

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:

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

The visual features $\mathbf{s}$ can be 2D feature leading to image-based VS (IBVS) or 3D features (such as the camera pose $\mathbf{r}$) leading to pose-based VS (PBVS). These visual features (points, lines, moments, contours, pose, etc) have thus to be selected and extracted from the images to control the robot DOF. In any case, it requires the knowledge or estimation of the interaction matrix $\mathbf{L_s}$ that links the temporal variation of visual features $\dot{\mathbf{s}}$ to the camera velocity $\mathbf{v}$ ($\dot{\mathbf{s}}(\mathbf{r}) = \mathbf{L_s}\mathbf{v}$).

To avoid the classical but error-prone extraction and tracking of geometrical features, DVS has been introduced, as already said in introduction. Considering the image as a whole, the set of features $\mathbf{s}$ becomes the image itself [7]: $\mathbf{s}(\mathbf{r}) = \mathbf{I}(\mathbf{r})$. The optimization process is then expressed as:

$$\widehat{\mathbf{r}} = \arg\min_{\mathbf{r}} \ \|\mathbf{I}(\mathbf{r}) - \mathbf{I}^*\| \tag{3}$$

where $\mathbf{I}(\mathbf{r})$ and $\mathbf{I}^*$ are respectively the image seen at the pose $\mathbf{r}$ and the desired image (both composed of N pixels). The main issue when dealing with DVS 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.

B. From DVS to CNN-based VS

In this paper, we propose to replace the DVS approach described above by a new scheme based on CNN. The network is trained to estimate the relative pose between the current and reference images. Given an image input $\mathbf{I}(\mathbf{r})$ and the reference image $\mathbf{I}_0$, let the output of the network be:

$$\mathbf{\Delta r_0} = net_{\mathbf{I}_0}(\mathbf{I}(\mathbf{r})) \tag{4}$$

with $\mathbf{\Delta r_0} = ({}^{c_0}\mathbf{t}_c, \theta\mathbf{u})$ the vector representation of the homogeneous matrix ${}^{c_0}\mathbf{T}_c$ that expresses the current camera frame with respect to the reference frame.

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 $\mathbf{\Delta}^*\mathbf{r}$ is easily obtained using ${}^{c*}\mathbf{T}_c = {}^{c_0}\mathbf{T}_{c^*}^{-1}\ {}^{c_0}\mathbf{T}_c$.

Expressing the cost function $\rho(.)$ with the Euclidean norm of the pose in Eq. (1), the minimization problem becomes

$$\widehat{\mathbf{r}} = \arg\min_{\mathbf{r}} \ \|\mathbf{\Delta}^*\mathbf{r}\| \tag{5}$$

which is known to present excellent properties [6]. Indeed, the corresponding control scheme belongs to PBVS, which is globally asymptotically stable (ie. the system converges whatever the initial and desired poses are), provided the estimated displacement $\mathbf{\Delta}^*\mathbf{r}$ is stable and correct enough. We recall that IBVS, and thus the schemes based on Eq. (2) and (5), 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 $\mathbf{\Delta}^*\mathbf{r}$ provided by the CNN, it is immediate to compute the camera velocity using a classical control law [6] :

$$\mathbf{v} = -\lambda \begin{pmatrix} {}^{c}\mathbf{R}_{c^*}\ {}^{c^*}\mathbf{t}_c \\ \theta\mathbf{u} \end{pmatrix} \tag{6}$$

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 VS

To implement the approach described above, we need to train a network by feeding it a significant number of images $\mathbf{I}_r$ for which we know the relative poses $\mathbf{\Delta r}$ with respect to the pose $\mathbf{r}_0$ corresponding to $\mathbf{I}_0$. 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 [19] 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 [17] 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 lower layers (basic image feature extractors) will perform similar functionality in our relative pose estimation. 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 the displacement between two poses, we choose an Euclidean cost function for network training, replacing the commonly used soft-max cost layer for classification, of the following form:

$$loss(\mathbf{I}) = ||\widehat{{}^{c_0}\mathbf{t}_c} - {}^{c_0}\mathbf{t}_c||_2 + \beta||\widehat{\theta\mathbf{u}} - \theta\mathbf{u}||_2 \qquad (7)$$

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

Starting from the trained AlexNet available for use with the Caffe library [15], the network was then fine-tuned. For this, a new scene specific dataset of 11 000 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.

III. Designing a Training Dataset

The design of the training dataset is the most critical step in a CNN since it affects the ability of the process to converge, as well as the precision and robustness of the end performances. As stated above we propose to fine-tune 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 real-world environments. Thousands of training configurations should be considered, which requires time when an actual robot has to be used. We now describe how simulated data allow us to generate a virtually unlimited amount of training configurations. 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.

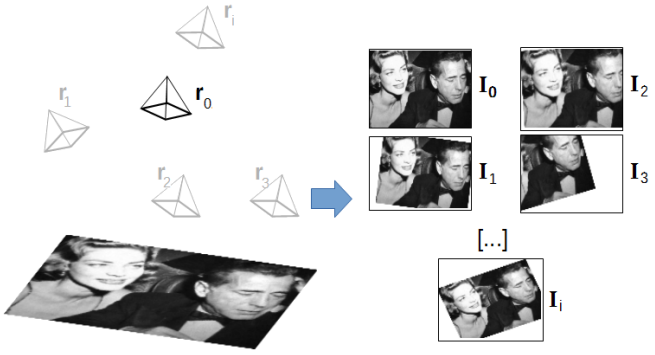

Fig. 2. 3D plane with the projected reference image and multiple virtual cameras to generate multiple views of the scene.

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 a large set of virtual cameras. Figure 2 illustrates the image $\mathbf{I_0}$ projected on a 3D plane from pose $\mathbf{r}_0$, the varying (virtual) camera poses and the images simulated from these poses. Since the 3D structure of the scene is a priori unknown, we rely on an image transfer technique based on homographies. This procedure generates datasets of thousands 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 [24].

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 $\mathbf{r}_0$, in order to get the camera characteristics and scene appearance
- create the elements of the training dataset, consisting of tuples $(\mathbf{r}_i, \mathbf{I}_i)$, 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 $\mathbf{r}_0$, 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 $\mathbf{r}_0$ is equal to 20cm, and the variances for the 6DOF Gaussian draw are (1cm, 1cm, 1cm, 10°, 10°, 20°), respectively for $(\mathrm{t}_x, \mathrm{t}_y, \mathrm{t}_z, \mathrm{r}_x, \mathrm{r}_y, \mathrm{r}_z)$.
- 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 $\mathbf{r}_0$ 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

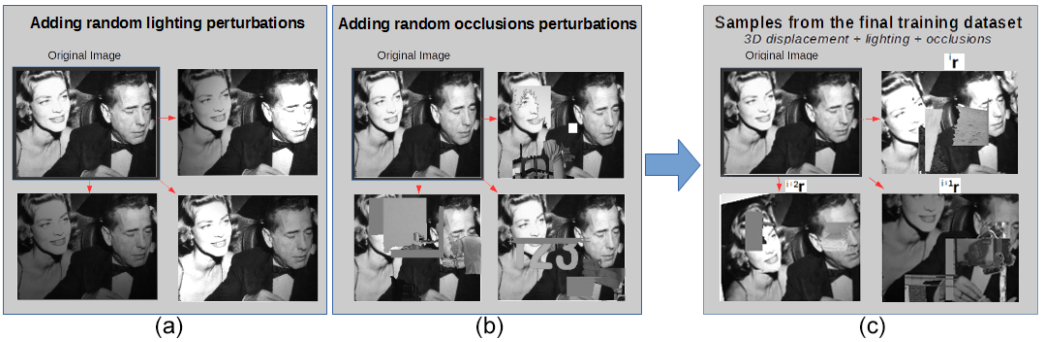

Fig. 3. Overall process to create a training set from the original input image: (a) examples after applying local illumination changes; (b) examples after adding super-pixels from random images as occlusions; (c) examples from the final dataset after applying all perturbations.

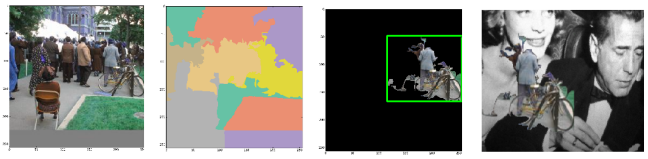

Fig. 4. Synthetic occlusion generation: on an arbitrary images in the *Label-Me* dataset (left) segmentation is performed. A segmented cluster is selected at random. It provides a coherent "occlusion" patch, which is merged with one of the dataset image (last image) and finally added to the training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

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 lighting variations, one can simply alter the pixel intensities by considering affine variation. Local lighting changes are more challenging and time-consuming rendering algorithms are required to obtain realistic synthetic images. 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):

$$\mathbf{I}_l(x,y) = \sum_{l=1}^{N_{lights}} \mathbf{I}(x,y) f_l(x,y) \tag{8}$$

Each 2D Gaussian in turn can be modelled as

$$f_l(x,y) = Ae^{-\left(\frac{(x-x_0)}{2\sigma_x^2}+\frac{(y-y_0)}{2\sigma_y^2}\right)} \tag{9}$$

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, 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 since the material and reflection properties are unknown. Our method will handle them as a sub-class of occlusions (see next paragraph).

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.

Clusters of pixels – representing a coherent part of an image – from other datasets are superimposed 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 [26] 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 [1] segmentation algorithm creating coherent pixel groups (implementation available in OpenCV)
- select a random cluster from the previous step, and insert it 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).

IV. Going further: Towards scene-agnostic CNN-based Visual Servoing

In this section we propose an extension of the previous method in order to obtain a network that can perform VS on scenes never seen during the training step.

A. Task definition

To operate within scenes never seen before, we use a similar network architecture as in the previous case, but train it for a new task that consists in computing the relative pose corresponding to two viewpoints of the same scene. This is made possible by training the network to associate a couple of correlated images of the same scene to a given camera displacement. Although the scale of the motion cannot be estimated without depth measurements, the direction of the displacement vector will be accurate and usable within a closed-loop control scheme.

The cost function for the network becomes:

$$loss(\mathbf{I}, \mathbf{I}^*) = ||\widehat{{}^{c^*}\mathbf{t}_c} - {}^{c^*}\mathbf{t}_c||_2 + \beta||\widehat{\theta\mathbf{u}} - \theta\mathbf{u}||_2 \qquad (10)$$

where $({}^{c^*}\mathbf{t}_c, \theta\mathbf{u})^\top$ is the relative position between the camera that acquires $\mathbf{I}^*$ and the camera that acquires $\mathbf{I}$. The important addition from (7) is that the optimization process now depends on two image inputs. The only correlation between these two images is that they both need to be a viewpoint of the same static scene.

B. Training dataset generation

Using the same set of tools as before, we propose a method to generate a dataset able to solve this new task from simulation tools and readily available datasets. This dataset will be used to fine tune a trained VGG network [28] (a deeper, more recent variation of the AlexNet network).

Each element of the training set is generated as follow:

- select a random image from a set of natural images, here from the Label-Me classification dataset used previously;
- set this image as texture of the 3D plane in the simulated scene;
- record two distinct viewpoints (chosen randomly) of this scene (network input);
- record the associated relative pose between these two viewpoints (data point label).

By repeating this process, an arbitrary number of annotated couple of viewpoints can be generated, creating an appropriate dataset for this new task. Perturbations can also be added in the same fashion as in the previous method.

Since this new task necessitates a more important reconfiguring of the trained VGG network (going from a single image input to a two-image input), we generated a bigger dataset compared to the previous training, going from 10k images to 100k images for training, which ran for 50 epochs.

V. Experimental results

Simulated experiments were first carried out to compare convergence area with respect to DVS methods. Real experiments with a 6 DOF robot were then considered. Section V.A and V.B concern the validation of the method introduced in Section II. Section V.C validates the extension presented in Section IV.

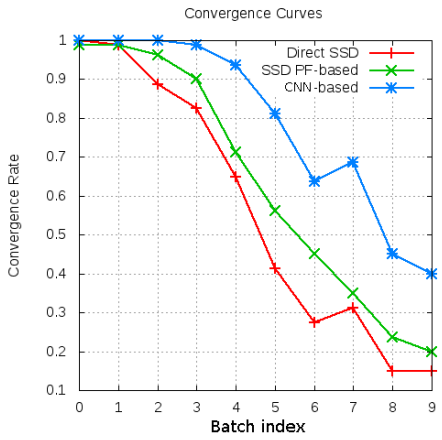

Fig. 5. The effects of the initial positioning offset (small offset for batch 0, large one with batch 9) with three DVS methods: DVS using SSD cost function (in red with +), a version using particle filter (green x) and our proposed CNN-based VS (blue *).

A. Experimental Simulations and Comparison with other DVS methods

In this section we compare the proposed method with the original DVS scheme [7] and with DVS based on a particle filter (PF) control law [3] that was recently introduced to increase the convergence domain of DVS.

Each scheme was evaluated by running a positioning task over 10 batches, each with 80 individual runs. The starting pose for each run was randomly offset (with a Gaussian distribution) from the desired pose $\mathbf{r}^*$. The magnitude of the offset was increased linearly from batch 0 to batch 9. This scene was simulated as illustrated in Figure 2, with the desired image $\mathbf{I}_0$ and the desired pose $\mathbf{r}_0$. The mean depth is 25cm and the offset pose was generated using the following variances (from batch 0 to batch 9): from 0 to 4cm for the X and Y translations, from 0 to 2cm for the depth, from 0 to 20° for rotations around the X and Y axes, from 0 to 50° for rotations around the Z axis.

The results of this benchmark are shown in Figure 5. It can be seen that the proposed approach has a larger convergence range with a significantly higher proportion of runs converging to the desired pose than with the other schemes for all starting poses with non-zero offsets.

B. Experimental Results on a 6 DOF Robot

1) Nominal Conditions: 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 and the task is, as usual, to position back the camera to a pose defined by a desired image. In this first set of experiments, the network (Alexnet here) was fine-tuned with a learning dataset built from an image $\mathbf{I_0}$ of the scene. Using the Caffe library, it trained for two hours on a K2200 GPU (2Go memory), corresponding to 50 epochs for a 11k images dataset.

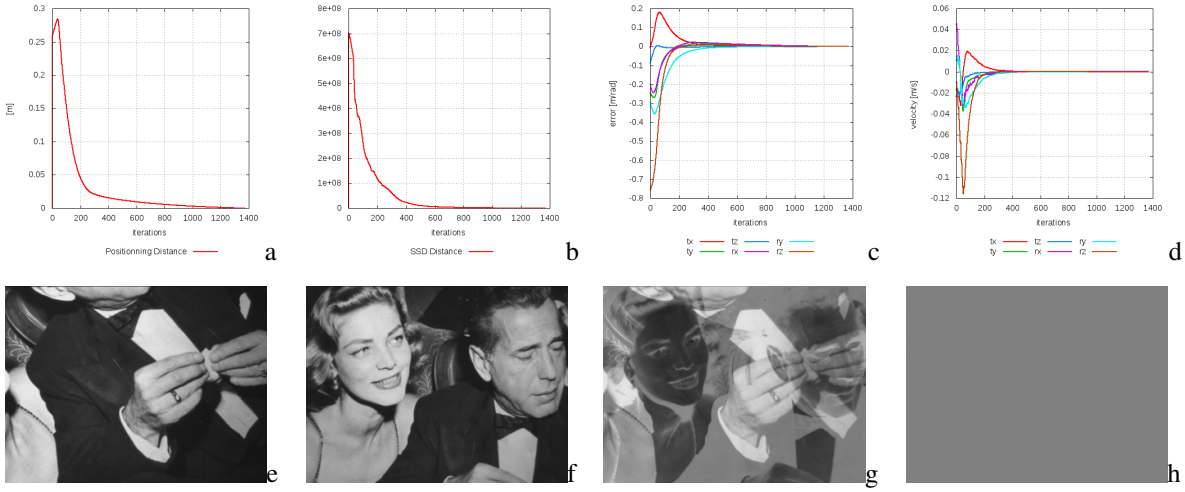

Fig. 6. CNN-based VS on a planar scene; (a) Positioning error; (b) SSD distance (not use in the control law, it should be zero at convergence when there is no occlusion and no lighting modification); (c) Translational and rotational errors; (d) Camera velocities in m/s and rad/s; (e) Image at initial pose; (f) Image at final pose $\mathbf{I}(\widehat{\mathbf{r}})$; (g) Image error $\mathbf{I}(\mathbf{r}) - \mathbf{I}^*$ at initial pose; (h) Image error $\mathbf{I}(\widehat{\mathbf{r}}) - \mathbf{I}^*$ at the final pose

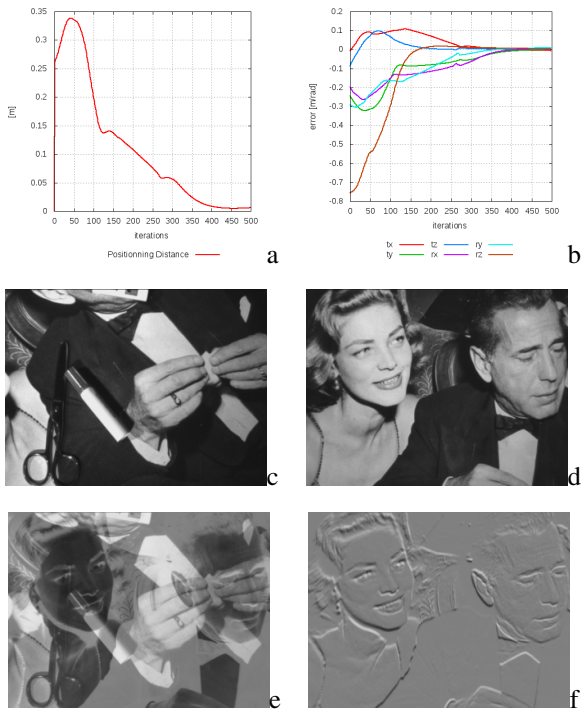

Fig. 7. CNN-based VS on a planar scene with various perturbations. (a) Positioning error; ; (b) Translational and rotational errors; (c) Image at initial pose; (d) Image at final pose $\mathbf{I}(\widehat{\mathbf{r}})$; (e) Image error $\mathbf{I}(\mathbf{r}) - \mathbf{I}^*$ at initial pose; (f) Image error $\mathbf{I}(\widehat{\mathbf{r}}) - \mathbf{I}^*$ at the final pose

In terms of pose initial offset, the robot has to perform a displacement given by $\mathbf{\Delta r_0} = ({}^{c_0}\mathbf{t}_c, \theta\mathbf{u})$ with ${}^{c_0}\mathbf{t}_c = (1\text{cm}, -24\text{cm}, -9\text{cm})$, $\theta\mathbf{u} = (-10°, -16°, -43°)$ while the distance between the camera and the planar scene is 80cm at the desired pose $\mathbf{r}^*$. Figure 6(f) shows the image at the final pose. Figure 6(h) shows the image error between the final and desired images. The training of the network with the 11,000 images was performed offline. Figures 6(a) through 6(d) show that our CNN-based VS converges without any noisy

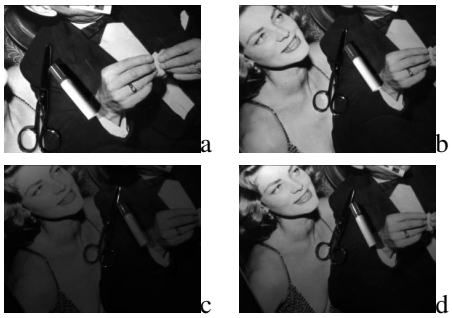

Fig. 8. Images collected during experiments on our 6DOF robot. Significant occlusions and variation in the lighting conditions can be seen.

nor oscillatory behaviours. Furthermore, the position of the system at the end of the servo 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 nor occlusions were added. These were introduced in the next experiment.

2) Dealing with perturbations: Light changes and occlusions: The same experiment is considered but additional light sources and external occlusions were added to test the robustness of our approach to perturbations. While the robot achieves the positioning task, the lighting conditions are modified (the intensities of three lights directed toward the scene are changed independently and arbitrarily). This results in global and local light changes, the apparition of specularities, etc. In addition, during the experiment, various objects are added, moved, and removed from the scene in order to create partial occlusions of the scene (see attached video).

Results of this experiment are depicted on Figure 7. 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

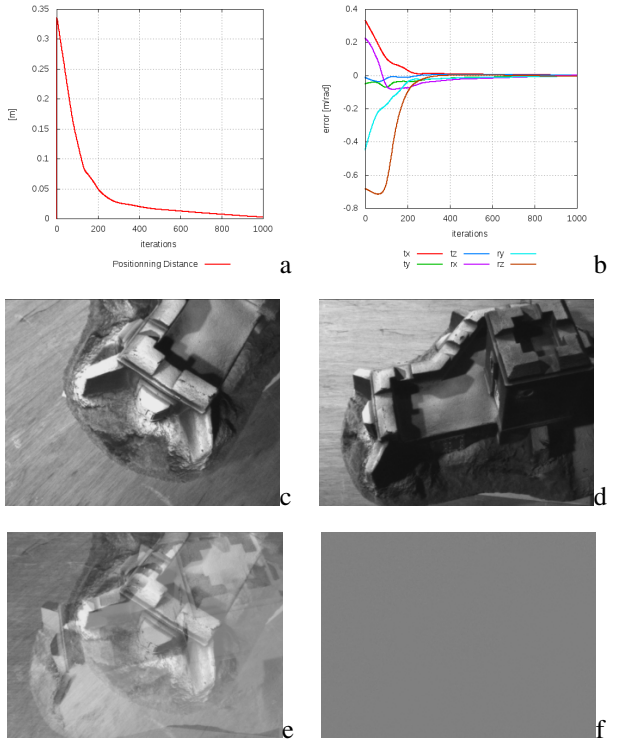

Fig. 9. CNN-based VS on a 3D scene: (a) Positioning error; (b) Translational and rotational errors; (c) Image at initial pose $\mathbf{I}(\mathbf{r}) = \mathbf{I_0}$; (d) Image at final pose $\mathbf{I}(\widehat{\mathbf{r}})$; (e) Image error $\mathbf{I}(\mathbf{r}) - \mathbf{I}^*$ at initial pose; (f) Image error $\mathbf{I}(\widehat{\mathbf{r}}) - \mathbf{I}^*$ at the final pose

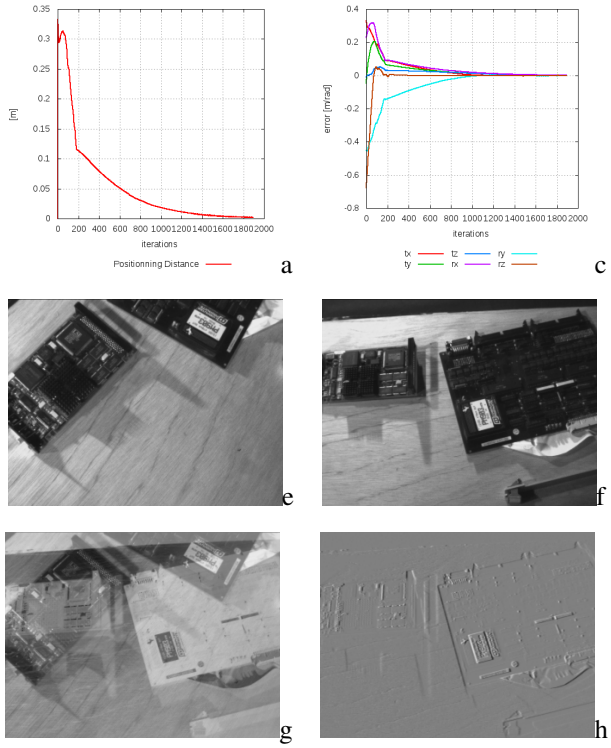

Fig. 10. Scene-agnostic CNN-based VS performed on a never-seen-at-training scene; (a) Positioning error; (b) SSD distance; (c) Translational and rotational errors; (d) Camera velocities in m/s and rad/s; (e) Image at initial pose $\mathbf{I_0}$; (f) Image at final pose $\mathbf{I}(\widehat{\mathbf{r}})$; (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

the perturbations (images acquired during the experiment are shown in Figure 8 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. Nevertheless, the camera reaches the desired pose once the perturbed conditions are removed.

3) Dealing with a 3D scene: In order to demonstrate further the robustness of the approach, we propose to consider in this experiment a large 3D object (Figure 9). It is important to note that the training of the network is still done *under a planar assumption*. In such a case, it is no more possible to keep a good match between the scene captured throughout the motion of the robot and the training dataset. Nevertheless, keeping the same training process, we took a single reference image $\mathbf{I_0}$ from the camera. As for the other experiments, we project this image on a 3D plane to generate the training dataset by moving a virtual camera around this 3D plane. Note that there is no knowledge about the 3D structure of the scene. Due to the loss of this 3D information, this creates a strong discrepancy between the training set and the real images that will be captured by the camera. The results of the experiment are given in Figure 9, with a final precision of less than half of a millimeter. One can note that despite the strong approximations in the training set, the robot is still able to converge. For this experiment, the robot had to perform a displacement given by $\mathbf{\Delta r_0} = ({}^{c_0}\mathbf{t}_c, \theta\mathbf{u})$: ${}^{c_0}\mathbf{t}_c = (33\text{cm}, -5\text{cm}, -1.2\text{cm})$, $\theta\mathbf{u} = (13°, -26°, -39°)$. The robot performs a large translational motion in order to change significantly the viewpoint of the scene throughout the experiment run. This way, the robot has to deal with strong discrepancies between the visual inputs and its training examples. Attached video shows a similar experiment featuring also large occlusions of the scene.

C. Results for a scene-agnostic trained CNN

In order to validate the extension toward a scene-agnostic method presented in Section IV, we performed the same experiment, this time on a scene the network has never seen at training step (see Fig. 10). This scene exhibits a large panel of challenging image characteristics, from low textures wooden areas, to soft shadows and slightly 3D and reflective surfaces on the electronic circuits.

We can see on Fig. 10 that the proposed method succeeds to position the robot within a few centimeters of the target pose, performing a large displacement, in both the image space and effector space. From this close pose (iteration 200), we switched to the classical DVS to achieve sub-millimetric accuracy. Using jointly these two methods allows for benefiting the best of each method: a broad convergence radius from the CNN-based method, and a very precise positioning around the optimum with the SSD-based method.

VI. Conclusion and Perspectives

We presented in this paper a new generic approach for robust VS using deep neural networks. We re-purpose and

fine-tune a pre-trained CNN by substituting the last layer with a new regression layer. The output of the CNN is then used to build a robust control law allowing precise re-positioning tasks. We presented a method to design and collect a synthetic learning dataset for fast fine-tuning of the network. This method allows to be robust to local illumination changes and occlusions. We also show that, with minor modifications, the proposed approach can be scene-agnostic, meaning that it can consider a scene that has never been considered in the learning dataset.

We demonstrated the validity and efficiency of our approach with experiments on a 6 DOF robot. The proposed method achieves positioning task within a millimeter accuracy. Furthermore, we have demonstrated that the proposed approach is robust to strong perturbations, such as lighting variations, occlusions, as well as unknown 3D geometry in the scene.

Future works will concern extending the CNN-based scene-agnostic visual servoing technique in order to increase its accuracy and robustness. In particular applying novel, more adequate network architectures, as well as investigating the integration of reinforcement learning in order to gain more control on the robotic motion dynamics during the positioning process.

REFERENCES

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE trans. on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[2] M. Bakthavatchalam, F. Chaumette, and E. Marchand. Photometric moments: New promising candidates for visual servoing. In *IEEE Int. Conf. on Robotics and Automation, ICRA'13*, pp. 5521–5526, 2013.

[3] 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, Daejeon, Korea, october 2016.

[4] Q. Bateux and E. Marchand. Histograms-based visual servoing. *IEEE RA-L*, 2(1):80–87, January 2017.

[5] Q. Bateux, E. Marchand, J. Leitner, F. Chaumette, and P. Corke. Visual servoing from deep neural networks. RSS Wk on New Frontiers for Deep Learning in Robotics, Boston, may 2017 (arXiv preprint arXiv:1705.08940).

[6] F. Chaumette and S. Hutchinson. Visual servo control, Part I: Basic approaches. *IEEE Robotics and Automation Magazine*, 13(4):82–90, December 2006.

[7] C. Collewet and E. Marchand. Photometric visual servoing. *IEEE Trans. on Robotics*, 27(4):828–834, August 2011.

[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, 2015.

[9] A. Dame and E. Marchand. Mutual information-based visual servoing. *IEEE Trans. on Robotics*, 27(5):958–969, October 2011.

[10] K. Deguchi. A direct interpretation of dynamic images with camera and object motions for vision guided robot control. *Int. Journal of Computer Vision*, 37(1):7–20, June 2000.

[11] 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*, 2016. arXiv preprint arXiv:1606.03798.

[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*, pp. 2366–2374, 2014.

[13] S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Trans. on Robotics and Automation*, 12(5):651–670, October 1996.

[14] M. Jagersand and R.C. Nelson. On-line estimation of visual-motor models using active vision. In *ARPA96*, pp. 677–682, 1996.

[15] 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 *ACM Int. Conf. on Multimedia, MM'14*, pp. 675–678, Orlando, FL, 2014.

[16] V. Kallem, M. Dewan, J.P. Swensen, G.D. Hager, and N.J. Cowan. Kernel-based visual servoing. In *IEEE/RSJ Int. Conf on Intelligent Robots and Systems, IROS 2007*, pp. 1975–1980, San Diego, USA, October 2007.

[17] S. Karayev, M. Trentacoste, H. Han, A. Agarwala, T. Darrell, A. Hertzmann, and H. Winnemoeller. Recognizing image style. In *British Machine Vision Conference*, 2013.

[18] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *IEEE Computer Vision and Pattern Recognition, CVPR'2015*, pp. 2938–2946, 2015.

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

[20] 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*, pp. 199–203, 1990.

[21] A.X. Lee, S. Levine, and P. Abbeel. Learning visual servoing with deep features and fitted q-iteration. *arXiv preprint arXiv:1703.11000*, 2017.

[22] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[23] E. Malis. Improving vision-based control using efficient second-order minimization techniques. In *IEEE Int. Conf. on Robotics and Automation, ICRA'04*, volume 2, pp. 1843–1848, New Orleans, April 2004.

[24] 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'16*, pp. 3406–3413, Stockholm, Sweden, May 2016.

[25] M. Rad and V. Lepetit. BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *IEEE Int. Conf. on Computer Vision*, Venice, Italia, October 2018.

[26] B. Russell, A. Torralba, K. Murphy, and W. Freeman. Labelme: a database and web-based tool for image annotation. *Int. Journal of Computer Vision*, 77(1-3):157–173, 2008.

[27] A. Saxena, H. Pandya, G. Kumar, A. Gaud, and K. M. Krishna. Exploring convolutional networks for end-to-end visual servoing. *IEEE Int. Conf. on Robotics and Automation, ICRA'17*, pp. 3817–3823, Singapore, may 2017.

[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR'15*, San Diego, CA, May 2015.

[29] G. Wells, C. Venaille, and C. Torras. Promising research vision-based robot positioning using neural networks. *Image and Vision Computing*, 14(10):715–732, 1996.

[30] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke. Towards vision-based deep reinforcement learning for robotic motion control. In *ACRA*, 2015.