Visual Feedback Control using CNN Based Architecture with Input Data Fusion

Adrian-Paul Botezatu
Dept. of Automatic Control and Applied Informatics
Gheorghe Asachi Technical University of Iasi
Iasi, Romania
adrian-paul.botezatu@academic.tuiasi.ro

Lavinia Ferariu
Dept. of Automatic Control and Applied Informatics
Gheorghe Asachi Technical University of Iasi
Iasi, Romania
lavinia-eugenia.ferariu@academic.tuiasi.ro

Adrian Burlacu
Dept. of Automatic Control and Applied Informatics
Gheorghe Asachi Technical University of Iasi
Iasi, Romania
adrian.burlacu@academic.tuiasi.ro

Teodor Sauciuc
Dept. of Automatic Control and Applied Informatics
Gheorghe Asachi Technical University of Iasi
Iasi, Romania
teodor-andrei.sauciuc@academic.tuiasi.ro

2022 26th International Conference on System Theory, Control and Computing (ICSTCC) | 978-1-6654-6746-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICSTCC55426.2022.9931843

***Abstract*—Visual servoing systems are designed to solve pose alignment problems by providing the necessary linear and angular velocities using data extracted from images. Among the difficulties encountered by the traditional visual servoing approaches, there are feature detection and tracking, camera calibration, scene complexity, and robotic system constraints. Part of these problems can be solved if Convolutional Neural Networks (CNNs) are added to a visual servoing architecture. The main advantage of CNNs is the capability of understanding both the overall structure and specific details of the images corresponding to the current and desired layouts. To take a step further the state-of-the-art architectures, in this paper, we show how extra input data can improve the visual servoing behaviour. The extra data result from maps of regions induced by the feature points' positions, without the necessity of employing tracking. The results obtained on relevant data sets show that the proposed input fusion-based CNN provides an improved approximation of the linear and angular visual servoing velocities.**

***Index Terms*—visual feedback control, convolutional neural networks, point features maps, early fusion.**

I. Introduction

Visual servoing deals with converting knowledge of the current and desired layout of a scene to a trajectory for the video sensor [1]. Subject to the video sensor kinematic relation with the robotic system, a visual servoing architecture can be designed with an eye-in-hand or eye-to-hand configuration. The research related to visual controllers followed different directions induced by either image processing approaches or control theory algorithms. Depending on the application area, a visual servoing architecture can be designed using visual features or the entire image as input, and can include classic PID (Proportional Integral Derivative) control techniques or advanced techniques such as model-based predictive control.

To overcome some of the problems that usually appear when the designed visual servoing architecture is put into practice, during the last years, Convolutional Neural Networks (CNNs) were included in visual feedback control strategies. The advantages brought to the visual servoing design are related to the capability of CNNs to learn significant features, without requiring preset feature extraction techniques. CNNs increase the degree of adaptability to fulfill visual servoing tasks without any prior information about the camera parameters or scene geometry.

Different neural architectures were configured from CNNs which were pre-trained for classification, to allow a fast learning process for specific visual servoing tasks [2]–[4]. In [5] the authors use FlowNet [4] to process depth data for approximating the camera pose. The authors from [6] employ AlexNet [2] and VGG16 [3] to design their architectures. These architectures are customised for visual feedback control and tackle real-time 6 degrees of freedom positioning tasks. Recently, [7] introduces DEFINet, which was designed as a Siamese architecture. The input data is processed by two CNNs which share neural layers. The deep features are sent to a regression block, the result being a prediction of the relative pose of an eye-to-hand camera.

In [8], the authors compare three CNN-based architectures for grasping detection. The best architecture uses a single branch, i.e. the input array is formed by concatenating on the depth of the two input images, and all six outputs are generated by a single regression block. Starting from this result, in this paper, we propose a CNN based on early data fusion, defined with a single branch. Early fusion consists in using additional available information to the input of the neural network [9], [10]. This information can be obtained using different approaches, e.g. from various modalities, by segmentation, filtering or feature points' detection, from the layers of other pre-trained CNNs, etc. [11].

The CNN architecture proposed in this paper uses relevant additional input information which is recovered via feature points' detection, without the necessity of tracking them. The approach still uses the entire current and final images as inputs, but it also exploits two maps indicating the areas around the strongest feature points which were detected for the initial

and final scenes. As the extra maps are inserted into the input arrays, they can be used by all the neural layers to focus on the most important parts of the scenes. Also, the deepest neural layers can integrate the information extracted from the neighbourhood of the feature points into the global context of the image, to achieve a refined understanding of the initial and final layouts. By receiving the maps of feature points both for the initial and the final images, the CNN can obtain associations between them, without an explicit tracking. To illustrate the value of the additional input maps, multiple neural models were built with a transfer of learning, for different testing scenes. The experimental evaluation done with a large data set [8] leads to the conclusion that regression performance is improved by the suggested early fusion-based CNN architecture. The main results of this study can be outlined as follows:

- The early fusion for the visual servoing task is considered between RGB images and additional maps of relevant feature points, which describe the main visual elements of the initial and final scenes; the proposed architecture can be easily configured starting from any model pre-trained for classification;
- Experimental evaluation done for simple and complex servoing scenarios, using different types of CNNs, different types of feature points and different configurations of the feature point neighbourhood show that the proposed early fusion improves the accuracy of the CNNs.

This paper is organized as follows: Section II presents some preliminaries on visual feedback control. Section III unveils the design of the proposed architecture, while the experimental results are discussed in Section IV. Section V is dedicated to conclusions.

II. Visual Feedback Control Systems

Visual feedback control refers to the use of image measurements and camera parameters to achieve a task-induced motion of a robotic system [12], [13]. The most cited state-of-the-art architectures - pose-based visual servoing (PBVS) and image-based visual servoing (IBVS) [14] - are designed to follow the next sequence of steps: extraction, tracking and matching of the visual features. The most used visual features are points, lines or image moments. Recently, different articles have proposed the use of the entire image as input in the visual feedback scheme. The PBVS architecture needs 3D information about the work scene and camera mode. This information is used to assess the position and orientation of the target. The IBVS approach makes use of the 2D image to compute the desired motion of the robotic system.

Both PBVS and IBVS have advantages and disadvantages when talking about putting theory into practice. The main disadvantage of PBVS is the necessity of knowing the camera calibration parameters. The IBVS approach requires advanced image processing algorithms to have a suitable set of image features, these being sensitive to external disturbances such as lighting. [1]

The research undertaken to improve visual servoing systems has as its main goals the increase of the precision, robustness to noise and trajectory smoothness. These goals can be achieved if machine learning techniques are added in the design phase of the visual feedback control. The main benefit of using machine learning is the convenience of automatically creating models from the initial-target paired images, without needing to pre-process these input images.

This work proposes a solution for visual feedback control based on CNNs with input data fusion. Details are provided in the next section. The goal is to integrate additional data based on point feature positions into the input arrays of a CNN. This allows the prediction of the camera velocities which should be transferred to the robotic systems, to perform the task of positioning an eye-in-hand setup from an initial configuration to a final one.

III. The Proposed Architecture

The visual servoing task consists of the approximation of the linear and angular velocities $\{v_x, v_y, v_z, \omega_x, \omega_y, \omega_z\}$, depending on a pair of initial and desired configurations of the scene. The output might be integrated as input data into the control law, in order to perform the movement from the current to the desired configuration. The main idea of this proposed solution is to combine the classical neural inputs with some additional information, to include more relevant data which can be used by the CNN in the regression task.

A. Design of CNN Architecture

The architecture of the proposed network is presented in Fig. 1. The input arrays were generated by the concatenation of the following items: the initial configuration described by I_1 (of size $M \times N \times 3$), the desired configuration described by I_2 (of size $M \times N \times 3$) and a pair of additional maps, P_1 and P_2, each map of size $M \times N \times 1$. As the concatenation is performed on depth, the resulting input arrays have the size $M \times N \times 8$.

Feature maps equivalent to P_1 and P_2 could be also extracted from I_1 and I_2 by means of convolutional layers. However, when the additional maps are set as inputs of the CNN, the information can be effectively processed by the whole series of neural layers. Also, the learning process can

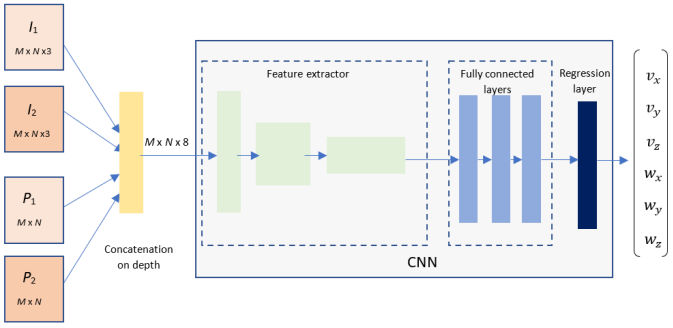

Fig. 1: The CNN using input data fusion

exploit these ready-to-use maps, without the need for training dedicated convolutional filters for extracting the same information. This will help CNN to focus on the most significant parts of the scenes while extracting deep features. The regions of interest are marked around the interest points, which are relevant for the regression task. Robustness to potential errors related to the detection of the feature points is sustained by the fact that the input arrays also include the original images, I_1 and I_2, from which CNN can retrieve necessary information.

The increased multidimensional array is fed into the feature extraction block made up of convolutional and pooling layers. The goal of this block is to produce a compact representation of the neural input, to facilitate the estimation of the linear and angular velocities. Any feature extractor and fully connected block design might be utilized, as long as it is suited to the size of the neural input and output. The most straightforward way is to start with a CNN model that has been pre-trained for image classification and then modify its architecture for the regression task. Proper initialization of the neural parameters allows a fast re-training with small training datasets.

The required modifications of the feature extraction block only consist of the adjustment of the input and the first convolutional layer for the desired number of input channels, as exemplified in Fig. 1. The new input layer should be simply generated for the input size $M \times N \times 8$, with the mean updated on each input channel. The first convolutional layer requires another modification because usually, pre-trained networks are designed for RGB input images, therefore the first convolutional layer includes filters with 3 input channels, $W_{F\times F\times 3}$, where F is the size of the filter. The filters of the new architecture, $W^*_{F\times F\times 8}$, can be initialized using the filters of the canonical CNNs. At this point, the learning transfer has special importance for a fast re-training, because an improper initialization of the parameters from the first convolutional layer alters the transfer of learning performed for the next layers. For the input channels associated with I_1 and I_2, the weights can be equal to $W_{F\times F\times 3}$. For the remaining two input channels, the configuration can take into account that the extra maps are generated in greyscale, and will contain higher intensities around the interest points and close to null intensities in the background. As consequence, the weights corresponding to additional maps can be obtained from $W_{F\times F\times 3}$ by using the conversion from RGB to grey:

$$W^*_F = 0.299W_R + 0.587W_G + 0.114W_B \qquad (1)$$

Here, W^*_F is the weight of the new filter, while W_R, W_G and W_B are the weights of a pre-trained filter W_{F*F*3} defined for the RGB channels. It is worth mentioning that these weights will be updated by the training procedure, in agreement with the regression task's goal.

The fully connected block can use the fully connected layers from the pre-trained model, except the last one, which includes 6 neurons approximating the linear and angular velocities. The last fully connected layer should not have activation functions incompatible with the range of the outputs, like rectified linear unit which does not allow negative output values.

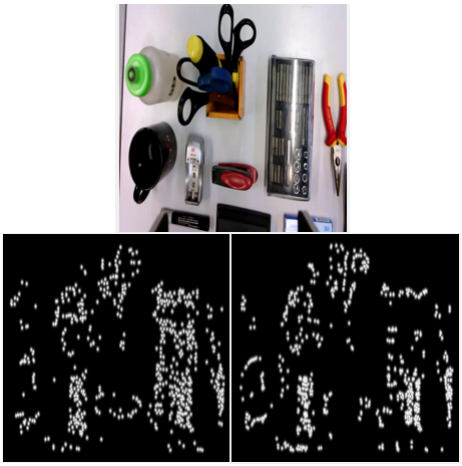

Fig. 2: Original image (top) and corresponding extra map - with SURF (bottom-left) and BRISK (bottom-right) detectors

As an exemplification, the experimental section considers different neural models derived according to this template. M_1 is built starting from the pre-trained AlexNet, hence all the above-mentioned modifications are mandatory. M_2 is built starting from the model introduced in [8]. For M_2, only the input layer and the first convolutional layer must be modified to fit 8 input channels; the last layers of the pre-trained model have been already customised for 6 outputs.

B. Generating the Maps of Feature Points

Our method proposes the integration of additional information into the input array, in order to obtain more relevant features for the regression task. Interest point operators are used to detect key locations from the image. They are selected to be less sensitive to scaling, rotation or disturbances of image quality. The investigations are done for two different extractors, to analyse the impact of the early fusion for different configurations of the additional maps. The SURF (Speed-Up Robust Features) [15] operator and BRISK (Binary Robust Invariant Scalable Keypoints) [16] operator were selected as examples, due to their robustness to noise and fast computation. [16] also recommends these two point extractors as well-established algorithms, with proven high performance.

A map with the detected SURF or BRISK points can provide information about the location of the objects. The SURF algorithm exploits a Hessian-matrix approximation, computed on integral images. Based on the scale-space principle, the method uses a Gaussian filter with an up-scaled size which is advantageous for numerical implementations. On the other hand, the BRISK method relies on circular sampling patterns from which it performs brightness comparisons to form a binary descriptor. Given the fact that the workflow is based on an initial and a final work scenes, the advantage of using such operators consists in the additional information about the variation of the interest points between these two images.

Algorithm 1 Map with SURF or BRISK points algorithm

Require: RGB image I (e.g., I_1 or I_2)
Step 1: Convert I to gray scale, resulting I_g.
Step 2: For I_g, detect the features points, p_i, with $i = 1, ..., n$.
Step 3: Create a Gaussian kernel, K, of size f_s, with standard deviation f_{sd}.
Step 4: For each point of interest, p_i, create a map I_{gi} having the same size as I_g; I_{gi} has non-zeros elements only around p_i, the neighborhood of p_i being defined by the kernel K centered in p_i.
Step 5: Create P, with the same size as I_g, by using for each pixel (x, y) the maximum corresponding value from the previously generated maps: $P(x, y) = max_{i=1,..,n} I_{gi}(x, y)$.
Ensure: Map of feature points, P (e.g., P_1 or P_2)

Algorithm 1 presents the main steps used for the configuration of the additional maps. In these maps, the detected points have higher grey-level intensities than the rest of the pixels. Given the fact that the initial and final input images are in RGB format, conversion to greyscale is necessary for applying feature point detection. With this conversion made, a pre-implemented algorithm for SURF or BRISK points detection is applied. The kernel that delimits the neighbourhood of the feature points is built with a Gaussian filter of size f_s and standard deviation f_{sd}. The final map is generated by mixing the neighbourhoods marked around all the feature points. Two examples for SURF and BRISK detectors are indicated in Fig. 2. The maps are obtained with a Gaussian filter of size f_s = 73 and a standard deviation f_{sd} = 5. Both detectors offer key points which highlight relevant parts of the objects; this can help the CNN understand the differences between the initial and final layouts.

IV. Experimental Results

For a proper assessment of the proposed method, early fusion is analysed for different neural architectures ($M1$ and $M2$), using different feature points (SURF and BRISK), and different configurations of the Gaussian kernel K. The reference is set by the models without early fusion, denoted $M1_6$ and $M2_6$, which have 6 inputs channels corresponding to the initial and final RGB images. The CNNs with early fusion receive two supplementary input maps depicting the points of interest, which leads to 8 input channels; these models are denoted as follows: $M1_{8S}$ and $M1_{8B}$ indicate the extension of $M1$ for the SURF and BRISK feature points, respectively, and $M2_{8S}$ and $M2_{8B}$ indicate the similar extensions for $M2$. The superscript indices specify distinct configurations of the kernel used for the construction of the extra maps.

A. Datasets and Training

The experimental investigations use a state-of-the-art data set [8], built with a Kinova Gen3 robotic manipulator. For a pertinent analysis, the experiments consider two working scenes of different complexity - with one or multiple objects (denoted 1 and 2, respectively). The dataset corresponding to a working scene includes 30000 pairs of initial and final images, each one associated with a target velocity vector. The design is done using 70% of the samples for training, 15% for validation and 15 % for testing. Two pairs of images corresponding to the adopted working scenes are exemplified in Fig. 3.

Fig. 3: Examples of training samples: initial (left) and final (right) layouts for the working scenes 2 (top) and 1 (bottom)

Training is performed via the Adaptive Moment Estimation (Adam) method, w. r. t. the minimisation of the Root Mean Squared Error (RMSE). The parameters of the training procedure are set as follows: 130 epochs, an initial learning rate of 0.0001, and a mini-batch size of 64. The neural model is assessed using the Mean Squared Output Errors (MSE) calculated for the training/ validation/ testing samples:

$$MSE = \frac{1}{6N} \sum_{k=1}^{N} \sum_{c=1}^{6} (y_c^t(k) - y_c^{net}(k))^2, \qquad (2)$$

where N indicates the cardinality of the training, validation or testing dataset, $y_c^t(k)$ and $y_c^{net}(k)$ denote the target and neural output, respectively, for the c^{th} output channel and the k^{th} example of the dataset.

B. Early Fusion Assessment

Table I and Table II present the MSE values achieved for the testing data set, for the two neural architectures, $M1$ and $M2$, configured with or without early fusion. The extra maps for the early fusion-based models are generated for the SURF or BRISK points, using a Gaussian kernel K defined for size f_s = 27 (which is about 4% of the minimum dimension of the original images), and a standard deviation f_{sd} = 5. Table I corresponds to the simple working scene, and Table II to the working scene with multiple objects. The experiments show that the extra maps help the CNNs focus on meaningful details. The two neural architectures, $M1$ and $M2$, provide

TABLE I: MSE for models with or without early fusion - Testing dataset, working scene 1

No.	CNN	v_x [m/s x 10^-5]	v_y [m/s x 10^-5]	v_z [m/s x 10^-5]	ω_x [○/s x 10^-5]	ω_y [○/s x 10^-5]	ω_z [○/s x 10^-5]
1	$M1_6$	2.9209	2.8154	4.0592	8.9356	6.3830	6.4336
2	$M1_{8S}$	0.8058	1.1116	2.1964	1.4215	0.8726	2.9742
3	$M2_6$	26.3546	23.8321	26.9956	38.4817	34.2442	39.3864
4	$M2_{8S}$	22.7901	20.8532	23.8162	33.2861	31.1923	37.1726

TABLE II: MSE for models with and without early fusion - Testing dataset, working scene 2

No.	CNN	v_x [m/s x 10^-5]	v_y [m/s x 10^-5]	v_z [m/s x 10^-5]	ω_x [○/s x 10^-5]	ω_y [○/s x 10^-5]	ω_z [○/s x 10^-5]
1	$M1_6$	2.0086	2.3068	5.1554	3.4767	2.3897	6.1174
2	$M1_{8S}$	1.5355	1.5428	2.4756	4.8917	2.7765	4.3635

TABLE III: Other configurations of the extra maps

No.	Config.	CNN	Detector	f_s	f_{sd}
1	C1	$M1^1_{8S}$	SURF	73	5
2	C2	$M1^2_{8S}$	SURF	73	15
3	C3	$M1^1_{8B}$	BRISK	73	5
4	C4	$M1^2_{8B}$	BRISK	73	15

more accurate results for the configurations with early fusion, for both working scenes. The improvement is visible for all the output channels. This outlines that the additional information from the input layers is relevant for the visual servoing task.

The benefit of early fusion is more significant for the first working scene. This scene has a larger background area than the working scene with multiple objects. As the feature points are mainly located on the objects, the extra maps usually indicate the background as irrelevant to the problem. The impact for the first working scene is thus bigger, also because larger areas can be discarded by the extra maps.

The results listed in Table I also show that the models based on $M1$ are better than those built from $M2$. The explanation mainly relies on the initialisation of the parameters. The parameters of $M2_{8S}$ and $M2_6$ were simply initialised by means of the Glorot algorithm, without transfer of learning from a pre-trained model. Differently, in the case of $M1_{8S}$ and $M1_6$, the parameters were initialised as explained in the previous section, via transfer of learning from the pre-trained AlexNet. As expected, the knowledge acquired by a pre-trained model is useful for effective training. In the case of $M2_{8S}$ and $M2_6$, a longer training stage might be useful, resulting that the model with input data fusion is more accurate than the canonical model defined with 6 input channels.

C. The Configuration of the Extra Maps

The impact of the early fusion is analysed for different configurations of the extra maps. The configurations are explained in Table III. They correspond to maps obtained with BRISK or SURF points, for different spreads of the kernel K. The higher the spread, the larger the neighbourhood which is marked by significant intensities around the points of interest. The extra maps are formed for the original images of size 1280 x 720,

TABLE IV: MSE for models with early fusion, for SURF and BRISK detectors - Testing dataset, working scene 2

No.	Config.	v_x [m/s x 10^-5]	v_y [m/s x 10^-5]	v_z [m/s x 10^-5]	ω_x [○/s x 10^-5]	ω_y [○/s x 10^-5]	ω_z [○/s x 10^-5]
1	C1	1.5887	1.3959	2.5151	4.4366	2.6301	4.3738
2	C2	1.6600	1.4653	2.2952	4.8211	3.0827	4.3262
3	C3	2.3339	2.1877	3.2787	7.9396	5.0341	5.5175
4	C4	2.3434	2.2612	3.3294	7.8815	5.0178	5.7616

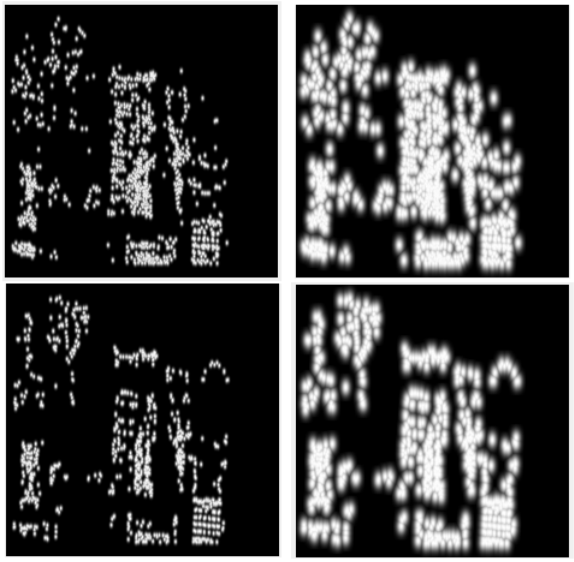

Fig. 4: Maps used in training for C1 (up-left), C2 (up-right), C3 (bottom-left) and C4 (bottom-right)

and then they are compacted to the resolution requested by the input layer of the CNN (e. g., 224 x 224 for M_1). As indicated in Table III, the Gaussian filter size is set to about 10% of the smallest image dimension. The experiments are done for two spreads, to investigate if it is more advantageous to mark larger or smaller neighbourhoods around the feature points. A large neighbourhood can be helpful if the detector does not provide enough key points. However, small neighbourhoods are useful to reduce the area recommended for further exploration, when all necessary key points are appropriately found. As mentioned before, the results listed in Table II are obtained with kernels of size f_s = 27 and f_{sd} = 5.

Fig. 4 exemplifies some maps of feature points obtained for all the four configurations indicated in Table III. The maps correspond to images captured for the second working scene. Both BRISK and SURF detectors depict relevant points of interest, however, according to Table IV, the locations of the SURF points are more relevant, on average, which translates into much lower MSE values for C1 and C2 vs. C3 and C4. The results show that the type of detector is more influential than the size of the neighbourhood marked around the points of interest. The differences between C1 vs. C2, or C3 vs. C4 are marginal, although the maps consider differently sized neighbourhoods. The results for C1 and C2 are also quite close to the results shown in Table II for $M1_{8S}$.

Fig. 5: The initial (left) and final (right) images for the testing sequence from Fig. 6

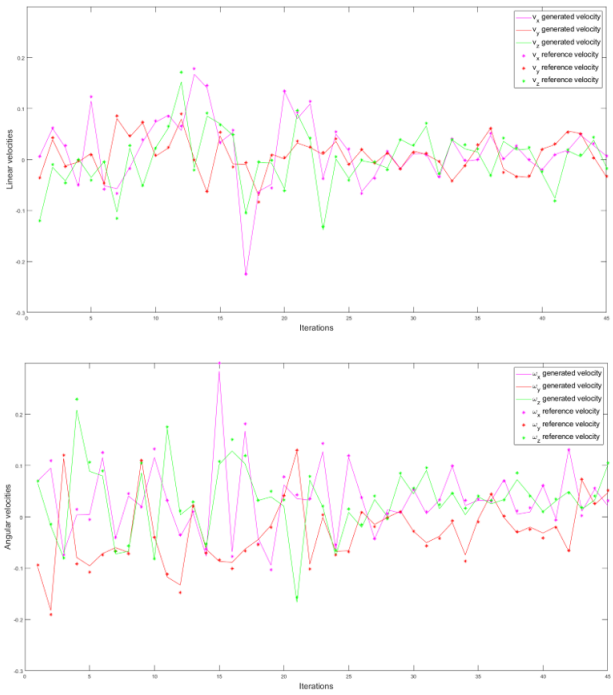

Fig. 6: Approximated velocities vs. reference velocities - for $M1_{8S}^{1}$: linear velocities - top; angular velocities - bottom

Additional details are indicated in Fig. 6, for the model $M1_{8S}^{1}$ defined according to C1. This figure shows the values of the linear and angular velocities for a sequence of images, generated with intermediary layouts between the initial and final images depicted in Fig. 5. The results highlight the good performance of the CNN when compared with the ground truth values represented with star shape markers.

V. Conclusions

This paper introduces CNNs with early fusion for visual servoing problems. Commonly, to compute the velocities corresponding to the movement of a 6 DOF effector's camera driven by visual feedback, the architectures admit as inputs two RGB images describing the initial and the final scenes. To increase accuracy, this study proposes to combine the raw RGB images with maps based on feature points. As the extra maps highlight significant areas of the working scenes, they implicitly guide the CNN in the extraction of deep features. The deep features thus embed the main differences between the final and initial images, without requesting an explicit tracking or matching. The experimental analysis was performed for two CNN architectures, for two different types of points of interest, namely SURF and BRISK, and for different neighbourhood sizes. For all the configurations of the extra maps, the models based on the proposed early fusion provide more accurate approximations, showing that the extra input information is really useful for the visual servoing problem. However, the impact of the early fusion is bigger for the maps obtained with SURF points, because it better illustrate the significant details of the scenes.

Future work is set to evaluate the behaviour of the proposed CNN-based architectures in real-time experiments for different types of robotic systems.

References

[1] F. Chaumette, S. Hutchinson, and P. Corke, "Visual servoing," in *Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2016, pp. 841–867.

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

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations*, 2015.

[4] A. Dosovitsky, P. Fischery, E. Ilg, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," *International Conference on Computer Vision*, pp. 2758–2766, 2015.

[5] A. Saxena, H.Pandya, G. Kumar, A. Gaud, and K. Krishna, "Exploring convolutional networks for end-to-end visual servoing," in *Proc. IEEE International Conference on Robotics and Automation*, Singapore, 2017, pp. 3817–3823.

[6] Q. Bateux, E. Marchand, J. Leitner, F. Chaumette, and P. Corke, "Training deep neural networks for visual servoing," in *Proc. IEEE International Conference on Robotics and Automation*, Brisbane, 2018, pp. 1–8.

[7] F. Tokuda, S. Arai, and K. Kosuge, "Convolutional neural network based visual servoing for eye-to-hand manipulator," *IEEE Access*, vol. 9, pp. 91 820–91 835, 2021.

[8] E. Ribeiro, R. Mendes, and V. Grassi, "Real-time deep learning approach to visual servo control and grasp detection for autonomous robotic manipulation," *Elsevier's Robotics and Autonomous Systems*, vol. 139, p. 103757, 2021.

[9] Y. Zhang, D. Sidibé, O. Morel, and F. Mériaudeau, "Deep multimodal fusion for semantic image segmentation: A survey," *Image and Vision Computing*, vol. 105, 2021.

[10] T. Zhou, S. Ruan, and S. Canu, "A review: Deep learning for medical image segmentation using multi-modality fusion," *Array*, vol. 3-4, pp. 1–11, 2019.

[11] L. Ferariu and E.-D. Neculau, "Fusing convolutional neural networks with segmentation for brain tumor classification," *25th International Conference on System Theory, Control and Computing (ICSTCC)*, 2021.

[12] G. Dong and Z. H. Zhu, "Position-based visual servo control of autonomous robotic manipulators," *Acta Astronautica*, vol. 115, pp. 291–302, 2015.

[13] M. L. Balter, A. I. Chen, T. J. Maguire, and M. L. Yarmush, "Adaptive kinematic control of a robotic venipuncture device based on stereo vision, ultrasound, and force guidance," *IEEE Transactions on Industrial Electronics*, vol. 64, pp. 1626–1635, 2017.

[14] F. Chaumette and S. Hutchinson, "Visual servo control Part I: Basic approaches," *IEEE Robotics & Automation Magazine*, vol. 13, pp. 82–90, 2006.

[15] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *Computer Vision and Image Understanding Journal*, 2008.

[16] S.Leutenegger, M.Chli, and R.Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *Proceedings of the IEEE International Conference ICCV*, 2011.