A Study on Learned Feature Maps Toward Direct Visual Servoing

Matthieu Quaccia, Antoine André, Yusuke Yoshiyasu, Guillaume Caron

A study on learned Feature Maps toward Direct Visual Servoing

Matthieu Quaccia[1,2], Antoine N. André[2], Yusuke Yoshiyasu[3], Guillaume Caron[2,4]

***Abstract*— Direct Visual Servoing (DVS) is a technique used in robotics and computer vision where visual information, typically obtained from camera pixels brightness, is directly used for controlling the motion of a robot. DVS is known for its ability to achieve accurate positioning, thanks to the redundancy of information all without the necessity to rely on geometric features.**

In this paper, we introduce a novel approach where pixel brightness is replaced with learned feature maps as the visual information for the servoing loop. The aim of this paper is to present a procedure to extract, transform and integrate deep neural networks feature maps toward replacing the brightness in a DVS control loop.

I. Introduction

A. Motivation

Visual Servoing (VS) refers to a control technique in robotics that uses visual feedback to guide the motion of a robotic system [1] and involves only visual sensors. Generally, the process aims to minimize a cost function built with the desired image and the current captured image.

The expression of this error depends on the way the information of the images pair are considered. VS can be categorized into two groups: Indirect VS (IVS) and Direct VS (DVS). IVS involves the extraction of features, such as keypoints [2] or contour features [3] for error calculation. In contrast, DVS relies on pixel brightness as direct input for robot control, without the need for feature extraction.

Photometric VS (PVS) [4], the first DVS, eliminates the need for feature tracking or matching processes. Recent variants of DVS have explored various image representation techniques resulting in expanded convergence domains, such as in [5] where Photometric Gaussian Mixtures are introduced as visual features for DVS. In [6], a novel approach to DVS is proposed by considering defocus as a way to optically smooth images without additional image processing. It demonstrates competitive convergence domains compared to the state-of-the-art methods, with larger domains in various scenarios, for a lower computational complexity.

In recent years, Convolutional Neural Networks (CNNs) have made computer vision tasks to progress significantly in various application fields, such as image classification [7] or 3D poses estimation [8]. The application of Deep Learning in robotics has also emerged, with CNNs being trained for tasks like grasp prediction [9], and complex positioning through reinforcement learning [10]. CNNs are also employed for estimating the pose difference between current and desired images, specifically in the category of Pose-based Visual Servoing (PBVS). PBVS relies on image information to estimate the camera pose, and CNNs play a crucial role in this process [11], [12].

Recently, new techniques of VS which combine the accuracy of DVS with the behavior and convergence of PBVS have emerged. In [13], the proposed method shapes a shared latent space and relies on multimodal information, resulting in accurate positioning and a broad convergence domain. In this space, the descriptions of camera poses and the associated image features are closely linked together.

B. Related works

A feature map, in the context of computer vision and image processing, refers to a spatial representation of specific dense features extracted from an input image. It can be thought of as a transformed version of the original image that highlights or encodes visual patterns or characteristics. It provides a compact and informative representation of relevant visual features inside an image.

Feature maps are typically obtained through various image processing techniques, such as CNNs which apply filters or convolutional operations to extract relevant features from the input image. In [14], the authors took advantage of the characteristics of the feature maps to implement a model selection technique for classification tasks. As they consider the spatial relationships between pixels, these feature maps might be useful for DVS purposes. Indeed, classical approaches consider pixel values that only represent intensity or color of discrete points in a grid and do not encode any spatial relationships.

C. Outline

The rest of this paper is organized as follows: Sect. II introduces a comparative study of neural network feature maps to determine the most suitable one for a VS application. Sect. III focuses on transforming the feature maps into 2D images, so that they can be integrated into a state-of-art DVS control loop. The final part of the article, Sect. IV involves conducting an experiment on a six degrees of freedom robotic arm robot to evaluate the error computation from feature maps. Sect. V summarizes the contributions of this work and presents possible directions for future researches.

II. Comparative study of neural networks feature maps for VS

A. Used Neural Networks presentation

In the following, we will study neural networks in order to choose which one could have interesting feature

[1]National Institute of Applied Sciences (INSA), Toulouse, France
[2]CNRS-AIST JRL (Joint Robotics Laboratory), IRL, Japan
[3]AIST AIRC, Japan
[4]Université de Picardie Jules Verne, MIS laboratory, France

maps for a VS application. Selecting the appropriate neural network for features extraction is critical, because finding relevant features will directly impact the effectiveness of the process. The feature maps prediction process must be efficient to achieve a desirable frequency in the servoing loop (60 hertz). Our investigation primarily concentrates on VGG16 (Visual Geometry Group [15]), and HRNet (High-Resolution Network [16]), both pretrained on the ImageNet dataset [17]. VGG16 offers a straightforward architecture and fast performance. On the other hand, HRNet aims to address the challenge of maintaining both high-resolution and high-level features throughout the network. We focus on these two networks mainly because they have both a simple architecture, and allow a simple feature extraction that can be easily transferred to the VS control loop.

First, we introduce two testing batches specially created to assess the performance of the networks. After that, we conduct an evaluation of the extracted feature maps to determine which one has a better spatial understanding, thereby making it more suitable for VS. In this context, "spatial understanding" refers to the capability of the feature maps to capture and represent the spatial relationships and arrangements of objects and features within an image.

B. Testing image batches creation

(a) First image of batch 1 (b) Last image of batch 1 (c) First image of batch 2 (d) Last image of batch 2

Fig. 1: First and last images of the two batches.

The two images batches consist of 50 frames targeting a planar scene (of a classical picture widely used in VS literature) and acquired at various orientations and positions using a Flir camera mounted on the end-effector of a robotic arm.

Fig. 1 shows the first and last images of the motion according to each batch. Concerning the batch 1, the sequence starts by showing the right part of the photo, and gradually moves along the width-axis of the photo to reveal the left part. For the second batch, the camera stream starts above the photo, and moves away (along z axis) while performing a pure rotation (around z axis).

C. Methodology

To study the performances and compare accurately the used neural networks, two different metrics that aim to compute the distance between descriptors of two image are put in practice. The first one is the Euclidean distance, expressed as:

$$D_e(X, Y) = |X - Y| \tag{1}$$

Where X and Y are vectors extracted at a specific location in the feature maps. The second used metric is the cosine distance:

$$D_c(X, Y) = \frac{X \cdot Y}{\|X\| \, \|Y\|} \tag{2}$$

Ideally, the pixel which has the lowest distance with the reference descriptor is located at the same scene position. As shown in Fig. 2, a descriptor is a feature vector at a specific location. All feature map's sizes are in the format $(width, height, channel)$, so each descriptor is channel-sized. We will plot the three points having the lowest distance with the input descriptor, and then evaluate if the resulting points correspond to the original location.

D. Performances evaluation

1) VGG16: The first neural network to be studied is VGG16. Fig. 3 highlights the architecture of this network built on 3 fully connected layers and 13 convolutional layers. We are focusing on the last convolution layers before each pooling layer (2, 5, 9, 13, and 17). These layers are outlined in red in Fig. 3. We are specifically focusing on these layers in order to study various levels of abstraction and different features representation. This approach allows us to explore how the network progressively captures and represents different complexities of features within the input data as it moves through the layers.

The results underscores that both too high-level and low-level layers struggle to capture essential spatial representations. This phenomenon is illustrated in Fig. 5, where we note that layers on these extremes yield unsatisfactory results in terms of spatial information capture. A trade-off can be found in layer 9, where a balance between resolution and spatial information can be achieved.

2) HRNet: The second neural network, HRNet, is then evaluated according to the same methodology (presented in Sect. II-C) and its architecture is summarized in Fig. 4. As mentioned in this section's introduction, HRNet allows multiscale information registration, while maintaining a high resolution representation. The Stem module in HRNet plays a crucial role in handling high-resolution images by capturing detailed information early in the network.

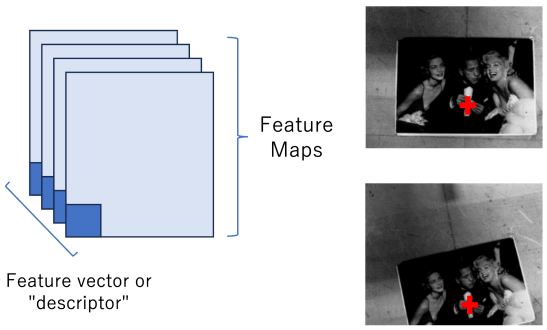


Fig. 2: Left: shape of the feature maps. To the right is an example of a reference point at the top, accompanied by the pixel with the lowest descriptor distance on a second image at the bottom.

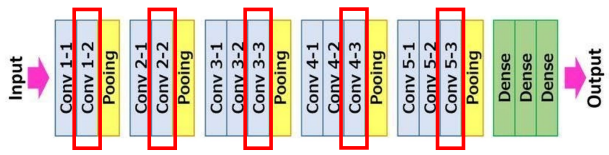


Fig. 3: VGG16 architecture. We study layers before each pooling layer (see the red rectangles).

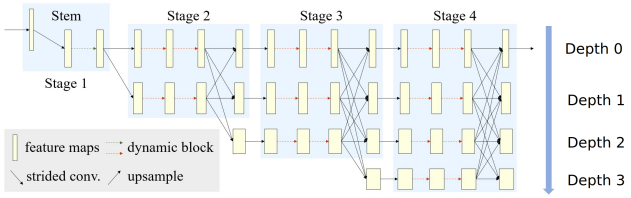


Fig. 4: HRNet architecture.

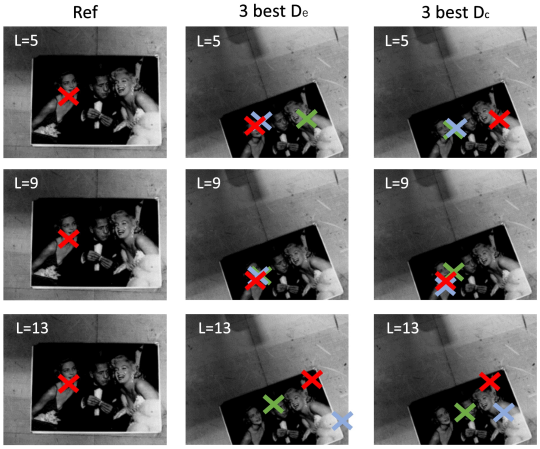


Fig. 5: Comparison examples between feature maps from various VGG16 layers. From left to right, we plot the reference point on the first image, the three best Euclidean distances on the second image, and the three best cosine distances on the second image. The best distances are represented by red, green, and blue crosses which respectively are the first, the second, and the third best distances. Real images are plotted instead of feature maps for the sake of understanding.

Using the same method as before, we see a similar pattern in the HRNet architecture. The first and fourth depths, which are respectively quite shallow and deep, struggle to capture important spatial information effectively. However, the second depth which is right in the middle, stands out in a positive way for HRNet, giving the best results for capturing the spatial data needed for DVS. This lines up with our earlier idea that a balanced depth is really important to get the best spatial representation. We can conclude that depth 2 is the best layer depth for a VS application, appearing to be a good trade-off between high resolution and consistent spatial information.

E. Final results

In this part, we will compare the performance of the layer 9 of VGG16 and the depth 2 in the HRNet architecture. Both feature maps are able to effectively capture the spatial representation of the images. We will compare these two feature maps in order to pick the suitable one for VS. We will use the same two images with the same reference point for the two architectures.

According to Fig. 6, we can say that the layer 9 of the VGG16 model provides the best feature maps for VS. We have observed that sometimes, higher resolution feature maps do not succeed to capture relevant visual data, and lower resolution feature maps lose too many spatial information.

Indeed, when comparing block 9 of VGG16 and depth 2 of HRNet, VGG16 outperforms HRNet in both scenarios. The results in Fig. 6 reveals that according to VGG16, the pixels having the lowest distances from the reference point were significantly closer to the reference point compared to HRNet. This suggests that VGG16 exhibits better performance in accurately capturing and matching keypoint descriptors, resulting in closer proximity between the matched points and the reference point.

We have now identified the most appropriate feature maps for a VS application. The next objective is to transform this 3-dimensional structure into a 2-dimensional representation. The detailed exploration of this transformation process is presented in the next section.

III. Feature maps to 2D image

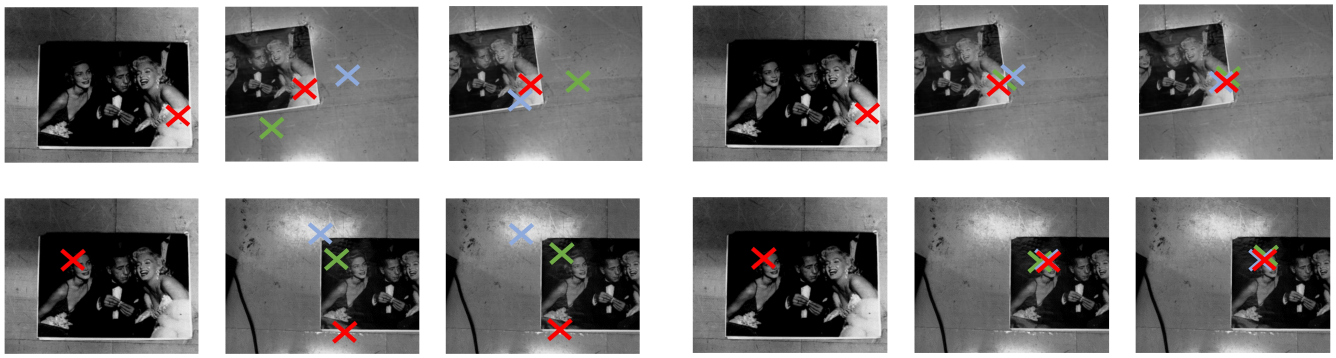
After selecting the best feature maps for VS, we have to transform this information into a 2D image that can be used in the servoing loop. This consists of a dimension reduction problem where all descriptors have to be represented as a single scalar. Several methods can achieve this task, such as Principal Component Analysis (PCA), or clustering. However, as the PCA method leads to a loss of information in the resulting image, we chose to rely on clustering for the transformation process.

The proposed method for clustering image feature maps combines a zone of interest detection and K-means clustering to identify meaningful clusters while excluding outlier points.

A. Clustering implementation

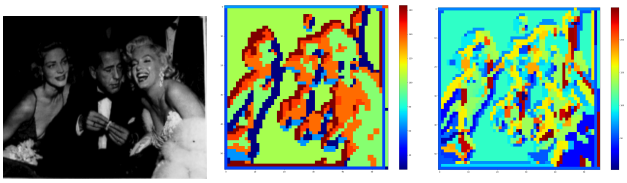
The clustering method used in this approach aims to fit a K-means model to the desired feature maps and predict clusters for the current image. The main objective is to group pixels with similar spatial information into the same cluster. We know that each descriptor vector represents a specific feature or characteristic of the image. The clustering algorithm assigns a cluster number to each descriptor, effectively grouping them based on their similarity. By employing the K-means algorithm, the feature maps are partitioned into K clusters, where K is a predefined parameter.

The number of cluster chosen with the K-means algorithm deeply affects the resulting kept spatial information. A first thought is that the higher the number of clusters, the more spatial information will be kept. But as highlighted in Fig. 7,

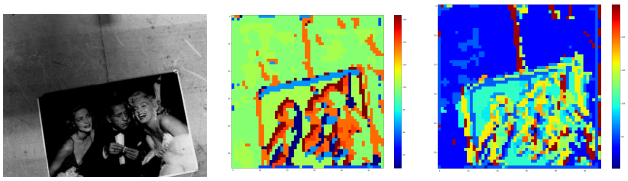


(c) HRNet depth 2 (d) VGG16 layer 9

Fig. 6: Comparison examples between feature maps from HRNet (depth 2) and VGG16 (layer 9).



(a) Desired image (c) K=10 (e) K=100



(b) Current image (d) K=10 (f) K=100

Fig. 7: From top to bottom, we can see at the left the desired and current images, at the center the desired and current clustered feature maps with K=10, and at the right, with K=100.

too many clusters can lead to a noisy clustered feature maps and over-segmentation. Striking the right balance in selecting the number of clusters is crucial to ensure that the clustering effectively groups pixels with similar spatial information without introducing excessive noise or unnecessary complexity. Our research suggests that 5 to 10 clusters is a good trade-off.

B. Noise reduction method

However, the K-means clustering algorithm can lead to the apparition of noise in the current image (as shown in Fig. 8). As this one is neither part of the desired, nor of the current image, it will result in under performing VS. To correct this issue, we decided to implement a method of region of interest detection with the help of SuperGlue network [18]. This supplemental step will then help to improve the accuracy and interpretability of the clustering results.

SuperGlue is a neural network-based method used for feature matching between two images. Given two images,

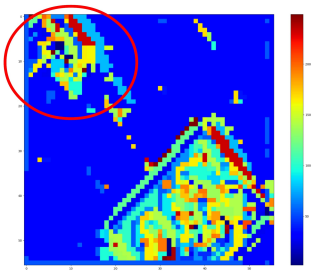


Fig. 8: Noise in the current image (100 clusters).

SuperGlue extracts features and finds the most likely correspondences between these features in both images. These correspondences are known as keypoint matches, where each keypoint in one image is matched to its corresponding point in the other image.

Once the keypoint matches are obtained from SuperGlue, a set of points in the current image that correspond to features in the reference image is obtained. To analyze the distribution and density of these points, we apply a Gaussian filter with

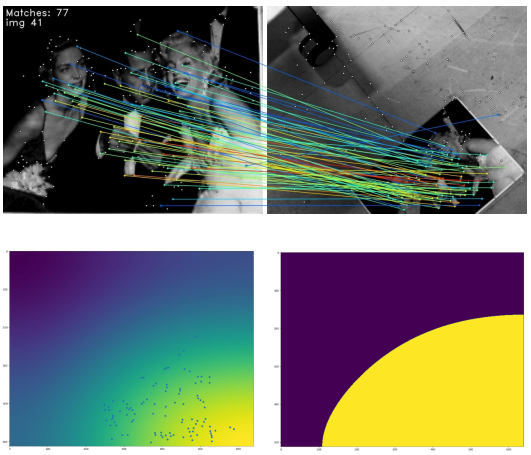
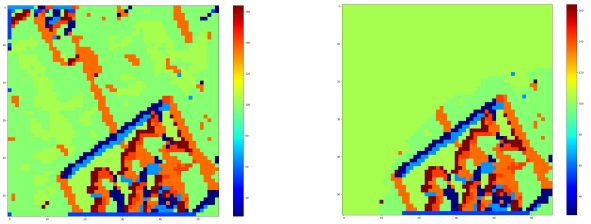


Fig. 9: Binary mask creation. Top: keypoints matches between the desired and current images. Bottom: keypoints density and the resulting binary mask.

a sigma value of 250 (helping to cover most of the area of interest in the reference image). This filtering process smooths the density values, providing a more continuous representation of the keypoints distribution. Fig. 9 highlights this method by showing the mask obtained after binarizing the Gaussian filter.

It is important to note that this noise removing filter is not meant to be applied during the whole VS process. Indeed, this noise removing mask is mainly useful when the difference between the reference and the desired image is important. When the two images are already very close, applying the filter becomes less relevant as it may not provide any significant improvement to the feature maps. In such cases, the filter might simply return the entire current image, leading to unnecessary extra processing without adding any valuable information.

In summary, the filter depicted in Fig. 10 has the potential to be a valuable pre-processing step in the VS control loop, helping to optimize the feature maps for more precise and efficient robot control.



(a) Clustered feature maps without mask

(b) Clustered feature maps with mask

Fig. 10: Comparison of clustered feature maps with and without noise removing mask (K=10).

IV. Experiments on a 6-DOF robotic arm

A. Experimental setup

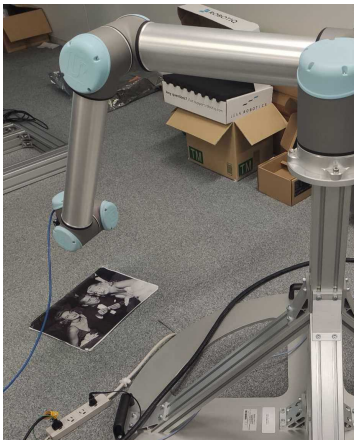


Fig. 11: Visual servoing setup. UR10, camera, planar target

The experiments in this study involves a 6-degree-of-freedom (DoF) Universal Robot 10 arm, with a Flir camera mounted on its end-effector. During the experiments, we use the Robot Operating System (ROS) framework [19]. ROS is a popular open-source middleware platform that provides tools and libraries for developing robotics software. For the visual feedback and image extraction, we used the C++ library provided by ViSP (Visual Servoing Platform) [20]. ViSP is an open-source library that offers robust and efficient implementations of various VS algorithms, making it well-suited for our experiments. In addition, we developed a Python node to publish the current feature maps acquired from the camera. The experimental setup used is showed on figure 11.

B. Toward Visual Servoing

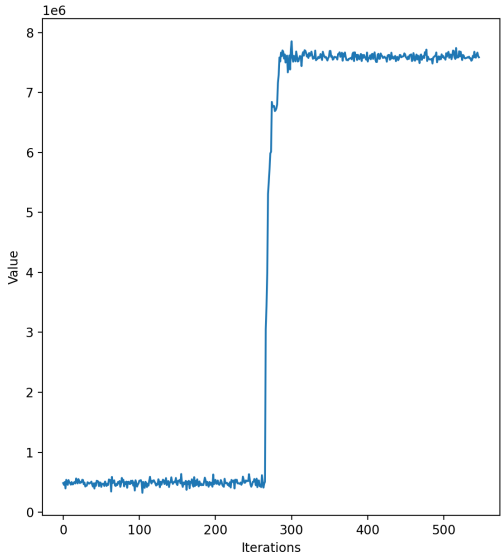


Fig. 12: Velocities sent to the robot and error.

This experiment aims to demonstrate the potential of feature maps to handle future tasks of VS. Therefore the robot is moved to see the sensitivity of the feature map-based cost with regard to the robot motion. After observing a phase where the robot remains static in an initial position, the end-effector (and the camera attached to it) is moved and remains static until the end of the experiment. The extraction of the feature maps allows to compute the cost between the feature maps acquired at the beginning of the experiment (used as a reference) and the current frame. This cost is visible in Fig. 12 and is computed as the sum of squared difference between the reference and the current feature map.

As expected, this cost is low when the robot stays at its desired pose, showing the ability to converge precisely when applied to VS. Furthermore, as the robot moves, we can observe an evolution of the cost that can be linked to the evolution in position of the robot. This information can then be used to feed a VS control law. Finally, the last part of the experiment shows a cost that remains constant, highlighting the stability of the cost evaluation through feature maps. We can also note that while remaining almost constant, the cost presents a non-negligible amount of noise that may disturb or decrease the performances of a VS law.

C. Discussion and possible amelioration

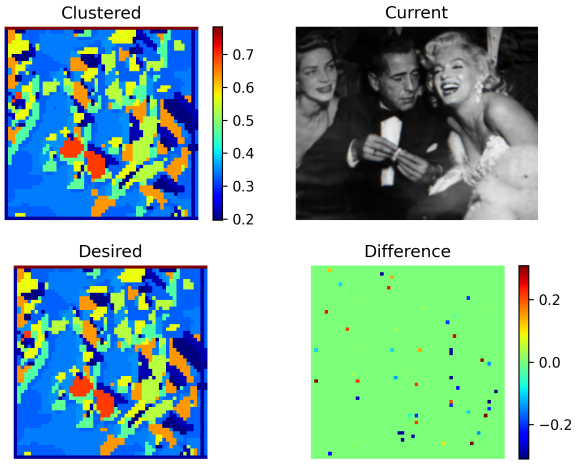


Fig. 13: Error between the desired and current clustered feature maps, when the desired and current image are acquired with the same camera pose.

During the experiments, we discovered that for an identical camera position, we got different clustered feature maps. As shown in Fig. 13, when the desired and current image are the same, the difference between the two resulting clustered feature maps is not null. The minor changes in pixel values (caused by illumination variation or camera noise) can propagate through the feature extraction process, leading to distinct feature maps. This instability of the feature maps can disturb the VS control law, leading to poor performances or even diverging cases.

To solve this problem, various approaches can be considered:

- Pre-processing techniques: Applying image preprocessing methods like histogram equalization or normalization can help to reduce the impact of luminosity changes before generating the feature maps.
- Noise reduction: Using noise reduction filters or denoising techniques can help minimize the effects of noise and small variations in the images.
- Adaptive clustering: Exploring alternative clustering algorithms that are more robust to minor variations could lead to more consistent and reliable feature maps.

V. Conclusion

In conclusion, our paper presents a visual servoing approach based on learned visual feature maps. We provided valuable insights about the extraction and dimensional reduction of feature maps, before studying its capability to be included in a DVS control law. The combination of neural networks feature maps, zone of interest detection, and K-means clustering allows the identification of meaningful clusters, excluding outliers.

This work shows the potential of the learned feature maps for DVS that will be able to replace the photometric feature by more meaningful information. However, there is still more work needed to truly achieve a complete servoing process and fully demonstrate the increased convergence domain, allowed by meaningful clusters extracted from the feature maps.

References

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

[2] E. Y. Puang, K. P. Tee, and W. Jing, "Kovis: Keypoint-based visual servoing with zero-shot sim-to-real transfer for robotics manipulation," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7527–7533, 2020.

[3] H. Wang, B. Yang, J. Wang, X. Liang, W. Chen, and Y.-H. Liu, "Adaptive visual servoing of contour features," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 2, pp. 811–822, 2018.

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

[5] N. Crombez, E. M. Mouaddib, G. Caron, and F. Chaumette, "Visual servoing with photometric gaussian mixtures as dense features," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 49–63, 2018.

[6] G. Caron, "Defocus-based direct visual servoing," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4056–4063, 2021.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[8] 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 *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 3828–3836.

[9] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 3406–3413.

[10] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, "Towards vision-based deep reinforcement learning for robotic motion control," *arXiv preprint arXiv:1511.03791*, 2015.

[11] Q. Bateux, E. Marchand, J. Leitner, F. Chaumette, and P. Corke, "Training deep neural networks for visual servoing," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3307–3314.

[12] S. Kamtikar, S. Marri, B. Walt, N. K. Uppalapati, G. Krishnan, and G. Chowdhary, "Visual servoing for pose control of soft continuum arm in a structured environment," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5504–5511, 2022.

[13] S. Felton, É. Fromont, and E. Marchand, "Deep metric learning for visual servoing: when pose and image meet in latent space," in *IEEE Int. Conf. on Robotics and Automation*, 2023.

[14] S. Mostafa, D. Mondal, M. Beck, C. Bidinosti, C. Henry, and I. Stavness, "Visualizing feature maps for model selection in convolutional neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 1362–1371.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[16] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang *et al.*, "Deep high-resolution representation learning for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 10, pp. 3349–3364, 2020.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[18] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4938–4947.

[19] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system." [Online]. Available: https://www.ros.org

[20] E. Marchand, F. Spindler, and F. Chaumette, "Visp for visual servoing: a generic software platform with a wide class of robot control skills," *IEEE Robotics and Automation Magazine*, vol. 12, no. 4, pp. 40–52, December 2005.