Dynamic Analysis and Modeling of DNN-based Visual Servoing Systems

Durdevic, Petar; Ortiz Arroyo, Daniel

Published in:
Intelligent Computing - Proceedings of the 2022 Computing Conference

DOI (link to publication from Publisher):
10.1007/978-3-031-10464-0_59

Publication date:
2022

Document Version
Early version, also known as pre-print

Link to publication from Aalborg University

Citation for published version (APA):
Durdevic, P., & Ortiz Arroyo, D. (2022). Dynamic Analysis and Modeling of DNN-based Visual Servoing Systems. In K. Arai (Ed.), *Intelligent Computing - Proceedings of the 2022 Computing Conference* (pp. 855-867). Springer. https://doi.org/10.1007/978-3-031-10464-0_59

Dynamic Analysis and Modeling of DNN-based Visual Servoing Systems

Petar Durdevic[1] and Daniel Ortiz-Arroyo[1]

Aalborg University, Esbjerg 6700, Denmark,
pdl@energy.aau.dk, doa@energy.aau.dk,
WWW home page: https://vbn.aau.dk/en/persons/125324,
https://vbn.aau.dk/en/persons/104617

Abstract. The integration of deep learning and control techniques has created robotic systems capable of implementing visual servoing and navigating autonomously in unknown environments. However, analyzing the effect that timing interactions between controllers and deep neural networks have, has received little attention in the literature. In this paper we describe a novel model that includes the effects that detection loss and inference latency have on the controller of a visual servoing system. To test our model we created a target tracking system consisting of a video camera mounted on a moving platform that tracks objects using deep neural networks.

Keywords: DNN, Visual-Servoing, Modeling, Control, Dynamics, Controlled Network Systems

1 Introduction

In [10] visual servoing is defined as a close-loop feed-back system that controls robot's movements by processing visual features extracted from camera images. Position-based servoing (PBVS) uses 2D image features together with the geometry of the 3D space to control a robot. Contrary, in image-based visual servoing (IBVS) only the 2D image features are used. A hybrid system combines the 2 approaches.

The survey on visual servoing in [12] describes a variety of approaches for grasping and planning in robotic manipulators. A more recent survey on visual servoing systems describe a variety of robotic systems that use classical and deep learnig-based techniques to perform visual servoing [18].

A form of visual serving is the visual tracking of objects with a robot equipped with a video camera. Visual tracking of objects can be done with or without a controller. When the target object is moving, the controller moves the camera mounted on a robot to track the object. In other tracking systems the target object is static but the camera on the moving robot, requires a controller to keep the object within its field of view (FOV). Lastly, in other tracking systems a controller is not required since the camera is static at a fixed position and the target object is moving. In this last case, objects will be tracked only as long as they remain within the static FOV of the camera.

State of the art computer vision systems rely on deep learning techniques to detect objects with high accuracy. Deep learning is a supervised machine learning technique whose goal is to minimize a loss function as expressed by the following equation:

$$\text{minimize } \frac{1}{n}\sum_{i=1}^{n} L(y_i, f_1(f_2(..f_m(\mathbf{w},\mathbf{t})))) \tag{1}$$

where L is the loss function, y_i are the testing labeled data of n images and $f_1(f_2(..f_m(\mathbf{w},\mathbf{t}))$ is a hierarchical deep neural network (DNN) of m layers. The DNN is trained on a dataset of size $|\mathbf{t}|$, with the goal of optimizing the internal weights $\mathbf{w}$ in the neurons and minimize the training error with the backpropagation algorithm [6,14,15]. The loss function used in a deep neural network depends on the task at hand and has a significant effect on performance. Some of the loss functions commonly used for regression tasks are the mean square error, cross-entropy, and Huber loss. For classification tasks, the categorical cross entropy, Hinge loss or Kullback Leibler Divergence are commonly used. In the case of object detection, where bounding box regression is required, the Intersection over Union (IoU) metric is commonly used as loss function.

The widespread use of deep learning techniques in visual servoing and target tracking systems is due to its significantly better performance, compared to the classical machine learning techniques. This is demonstrated by the fact that Deep Neural Network (DNN) models have consistently won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), in the categories of object detection and recognition [22] since 2012.

This paper presents the modeling and analysis of a visual servoing system for tracking objects that includes a DNN sensor in its feedback control loop. We call our sensor Deep Neural Network Visual Servoing (DVS) sensor. Our experimental setup is an object tracking system that uses a servo mechanism to move the camera and keep the object within the field of vision of the camera. The effect of inference latency and detection loss on the system performance is studied and a dynamic model is defined. In addition, we discuss some potential control strategies. To our knowledge no other work has performed this type of analysis on DNN-based visual servoing or tracking systems.

The paper is organized as follows: section 2 describes relevant related work on the topic. Section 3 describes the two main factors that affect the timing interaction between the DNN and the controller in our system. 4 describes the experimental set-up used, section 5 describes the model, section 6 discusses some potential control solutions and section 7 concludes the paper.

2 Related Work

In recent years, many visual servoing and tracking systems based on deep neural networks and other machine learning techniques have been proposed, An example is [11], in which the authors use the Interacting Multiple Models (IMM) algorithm to fuse several linear dynamic models of the target object. Additionally, the gain of an Extended Kalman Filter is automatically adjusted to minimize tracking errors.

In a similar work, in [21], a Kalman filter is also used for object tracking, but in this case its noise covariance matrices are fine tuned through the Particle Swarm Optimization algorithm. Lastly, a face tracking algorithm is described in [29], combining a Kalman filter used as a visual state estimator and an echo state network-based self-tuning algorithm to create a robust face tracking system.

In all previous visual tracking systems the camera is static. Contrarily, visual servoing tracking systems include a controller to move the robotic platform in which the camera is mounted, to follow or reach an object. Examples of visual servoing tracking systems are drones equipped with DNNs that track and follow forest trails autonomously as is described in [5].

Other examples of DNN-based visual servoing systems, are drones capable of detecting and tracking wind turbines to navigate towards them. An example of such a system is [20]. Another recent example is [4], where moving objects are tracked and followed by a ground robot. In this last case, the tracking system employs MobileNet [9], a DNN based on the Single Shot Multibox Detector (SSD) [16] architecture that detects objects and a PI controller that controls robot's movements to keeping the target object within the FOV of the camera and at a safe distance.

Lastly, using a completely different approach, in end-to-end systems the controller is completely eliminated by a DNN that has learned to mapping directly, image pixels to steering wheel commands, as is described in [2] using supervised machine learning. A similar system, Dronet described in [17], is capable of learning the steering angle and a collision probability, by using training data collected from driving cars and bicycles.

3 Effect of DNNs on Controllers

In all applications on visual servoing and visual tracking described in previous section, the DNNs continuously perform inferences in real-time. However, inferencing is a computationally expensive task and therefore, DNNs may introduce delays in the feedback control loop that may cause system's instability. The delay could be substantially reduced by optimizing DNN's architecture or using faster parallel processing hardware. For instance, optimized DNNs such as MobileNet V3-Small have a latency of 43 ms with an accuracy of 16.1 mAP [8] on mobile devices. TPUs and Jetson Nano Graphical Processing Units (GPU) boards can provide latencies from 2.9 to 18 ms when performing inferences with MobileNet V2 [7]. In spite of these low delays, the latency due to inferences performed in real time may still be significant even. This is because of the limited computing power in the host processor that additionally to the DNN, should manage the multiple tasks in kernel's scheduler and the communication to send and receive commands that will be used to move and localize the robot at a certain position.

The effect of latencies in a DNN-based visual servoing system, is similar to the effect that random delays have on networked control systems. In this last case, the delays happen when the sensors send their data to the controller through a communication network of limited bandwidth as is described in [26] and [30]. One possible solution to the time delay problem in networked control systems is to use buffers, however when the delay is large the control system may be responding to past events causing system's instability [19].

Latencies are not the only problem in DNN-based visual servoing systems. DNNs may be also unable of detecting objects, occasionally. This may happen when there are poor lighting conditions or the robot or target object are moving fast. In these conditions, no output detection will be produced by the DNN. We call this effect *detection loss*. One way to alleviate this problem is by using Kalman filters to estimate the most likely position of the target tracking object when a detection lost has been detected. However, the limitation of this approach is that if detection loss is significant, the Kalman filter may be unable of tracking an object.

The detection loss effect in DNNs is similar to random packet loss in networked control systems. In these systems, data packets sent by the sensors or the controller may be lost in the network due to noise or traffic congestion in the routers. The network protocols used, either TCP or UDP, have different effects on the control system's behavior since lost packets may or may not be sent again.

Including the effect of DNN's latency and detection loss in the modeling of a visual servoing system is a challenging problem since this makes the system no longer time-invariant.

4 Experimental Setup

We created an experimental visual servoing system that includes our DVS sensor and a control system. Its function is to detect and keep track of an object, so that when the servo motor that has the camera mounted moves, the camera rotates to keep track of an object. The object we have chosen for our experiments is a human face. Figure 1 shows the main components of our experimental setup.

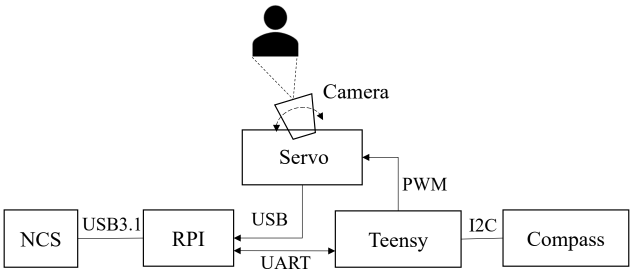

Fig. 1: Experimental setup of the visual servoing system

As figure shows, an Intel's Neural Compute Stick 2 (NCS) is attached to a Rasberry PI v3 (RPI) board through an USB 3.1 port. The Logitech 920 video camera sends images to the RPI that are processed by the DNN running in the NCS to detect objects. The Teensy v3.6 board is connected through I2C bus to a servo motor (HD-1800A - analog servo v.2) with the video camera attached. As the servo and the camera move, the NCS detects changes in the position of the target object within the image and sends the updated position to the controller running on the RPI. The controller in turn, sends a PWM signal to the servo motor attached to the Teensy board that rotates the camera in the x coordinate an angle ψ, to keep the target object within the field of vision of the camera. The IMU BDO055 compass, measures the three axis orientation data of the servo and camera. This data is used to compute the rotation angle of the camera and compare it with the calculated angle using the DVS sensor's output data as is described in the next section.

4.1 DNN and Communication Protocol

In our experiments we used a pretrained DNN that has MobileNet V1 [9] as backbone. MobileNet was designed to have a small size and low latency. Its low latency is mainly achieved by using more efficient depthwise convolutions rather than standard convolutions in the first layers of the network. Contrary to standard convolutions, in depthwise convolutions, instead of applying the convolution filters and then combining the inputs in a single layer, the filtering and combination is performed in two layers. First, a single filter is used for each input channel and then pointwise convolution is applied to combine the outputs.

MobileNet was pretrained to detect faces in images, producing as its output the position of the bounding box around a face and a confidence value.

The image detection output of the DVS has the following data format:

$$DVS_{out} = [i_{id}, l, c, x_{min}, y_{min}, x_{max}, y_{max}]$$

where i_{id} is the image id, l is the label, c is the confidence value, x_{min}, y_{min} are coordinates of the top left bounding box corner, and x_{max}, y_{max} are the coordinates of the bottom right bounding box corner.

To communicate data from the MobileNet DNN, running on Intel's NCS on RPI board, to the Teensy board, we created a simple communication protocol that sends data packets containing bounding box data and the detection latency of the network trained to detect faces, that achieved a confidence value larger than a threshold set arbitrarily to 0.7. The data packets sent have the following data format:

$$[\Delta t_{d_p}, DVS_{out}]$$

Where Δt_{d_p} is the detection data arrival time difference at the controller node and DVS_{out} is the data output of the network. The data packets are sent to the Teensy board through the UART interface, when a face is detected on an image.

With the DVS_{out} values we compute the width, height (w, h), and the center coordinates of the bounding box $[u, v]^T$

$$\begin{aligned} w &= x_{min} - x_{max} \\ h &= y_{min} - y_{max} \end{aligned} \tag{2}$$

$$\begin{aligned} u &= x_{min} + \frac{w}{2} \\ v &= y_{min} + \frac{h}{2} \end{aligned} \tag{3}$$

Using the center of the bounding box u value in Equation 3, the rotation angle for the camera in the x direction is calculated in equation 4. This angle is used to keep track of an object located within a bounding box.

$$\psi_{DVS} = atan\left(\frac{u - u_c}{f}\right) \tag{4}$$

where u_c is the center of the image frame and f is the focal length.

To calculate the rotation angle we need camera's intrinsec parameter focal length. The Logitech 920 video camera used in our experiments has the intrinsec parameters shown in Table 1.

Intrinsic parameters	Value	unit
Resolution	[672,384]	pixels
Focal Length	3.67	mm
Pixel Size	3.98	μm

Table 1: Camera's intrinsic parameters

5 Model

A visual servo system can be represented by the block diagram in figure 2 where the output is generated by a DVS sensor. The sensor is perturbed by noise v_k which is added to the DVS signal $\tilde{y}_k$ that is used as the feedback signal to a controller. This signal may suffer from detection losses, an effect similar to dropout of packages γ_k in networked control systems. Additionally, the signal may arrive with a delay τ_k. The controller in this case can be of any type.

In the next sections we discuss how to model the detection loss and the delay.

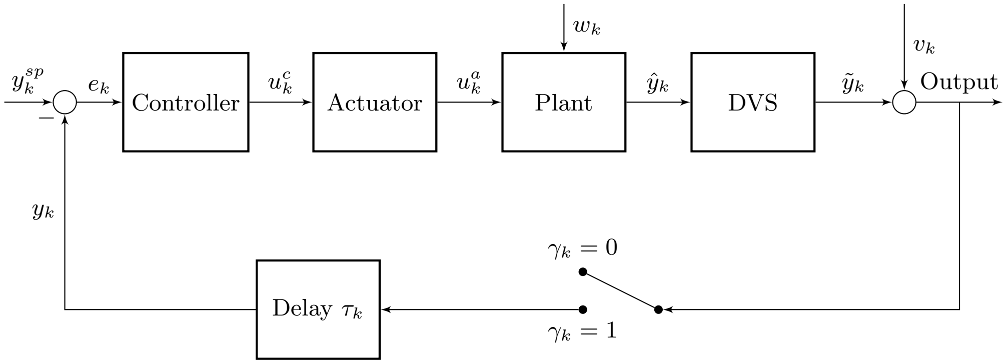

Fig. 2: Block diagram of the considered system, we can simplify this and remove the reference for now, as this part is not considered in this paper

5.1 Definition of the System with Detection Loss

A stochastic plant can be described as a discrete time linear dynamic system by the equation 5 [26]:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + w_k \\ y_k &= \gamma_k(Cx_k + v_k) \end{aligned} \tag{5}$$

and the controller's output in discrete time

$$u_k = -Kx_k \tag{6}$$

where $x_k \in \mathbb{R}^n$ is the state vector, $u_k \in \mathbb{R}^m$ is the control input, w_k is the white noise process, and v_k is the measurement error and K is the control gain.

γ_k is formally known as the package loss as described in [24–28], and is is defined as a Bernoulli random variable.

In our case it represents the detection loss, and it is defined in equation 7

$$\gamma_k = \begin{cases} 1 & if \text{ detection data arrives before or at time } t,\ t \le k, \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where the arrival time is defined as t_k and k is the sample number with a smapling period t_s, and thus for every detection the DVS sensor sends a packet y_k. That is:

$$y_k = \begin{cases} Cx_k + v_k & \text{if the object is detected} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

this is the main difference from the formulation in [26], where:

$$y_k = \begin{cases} Cx_k + v_k & \text{if the packet is delivered} \\ v_k & \text{otherwise} \end{cases} \tag{9}$$

i.e. if the packet is not delivered the output is pure noise.

The detection $\mathbf{D}$, is defined as:

$$\mathbf{D} = \begin{cases} 1 & \text{if } p_{\mathbf{d}} > \zeta \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where $p_{\mathbf{d}}$ is the probability of detecting an object and ζ is a threshold. The detection success, i.e. $\gamma_k = 1$, is dependent on the DNN's inferencing process, which is difficult to model as it depends on a multitude of identifiable and unidentifiable sources. Some of these are: type of the DNN, amount of data used for training, size of the object within an image, current lighting conditions, use of acceleration hardware and amount of memory available in the computing device, among others.

Assumption 1. *If the object cannot be detected for $k = \{t, \infty\}$ by the DNN, we will have the following condition: $\gamma_k = 0, \; for \; k = \{t, \infty\}$*

Typically this is considered as a sensor failure, but in the case of the DVS this represents the inability of the DNN to detect an object even though the system continues working.

The DVS's stochastic and dynamic properties have been used in our experimental set-up, described in section 4. The experimental results are shown in figure 3. In this figure, the top plot shows the reference signal for the servo $[u]$, the angle of the camera measured with the IMU $[\psi]$, the object location in the image plane measured by the DNN [DNN], the data is normalized. The middle plot shows the computed translational velocity of the object, relative to the camera. The bottom plot shows the detection data arrival time difference at the controller node $[\Delta t_{d_p}]$.

From the results in figure 3, we can observe that the detection loss occurs more frequently as the relative object's velocity increases and thus it becomes more difficult to detect. The unknown properties of γ represent a great challenge to the feedback control structure, as the loss of feedback signal might bring the system to instability.

5.2 Delay

In addition to the detection loss, the system shown in figure 2, suffers from a detection delay. The detection delay at time k, τ_k, represents the delay or latency of the DVS. τ_k comprises multiple delay sources, the main of which are:

$$\tau_k = \tau_{\mathrm{os}} + \tau_{\mathrm{if}} + \tau_{\mathrm{cp}} \tag{11}$$

where τ_{os} is the delay caused by the operating system scheduling, τ_{if} is the delay caused by the inference process within the DNN, τ_{ct} is the delay caused by the communication protocol. Thus τ_k represents the time from when the object is observed to the time when its detection is outputted from the DVS and

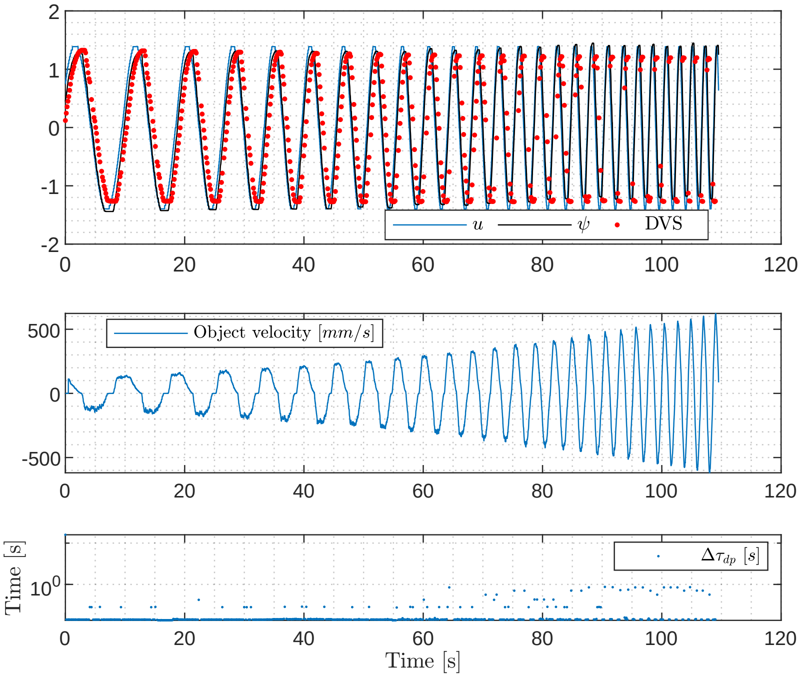

Fig. 3: Experimental results, analysis of the DVS sensor.

consequently this is analogous to the bandwidth of the DVS. In the experimental set-up, We have observed in our system that the following relationship holds:

$$\tau_{\text{if}} > \tau_{\text{os}} + \tau_{\text{cp}} \tag{12}$$

meaning that inference time is the largest latency. This relationship is highly dependable on the network type and size, and on the hardware and communication protocol. A more powerful tensor processing unit (TPU) could potentially invert this relationship.

Another important aspect is that the total delay τ_k is dependent on the detection loss γ_k, similarly as was stated in [24], having the following relationship:

$$\tau_k = \begin{cases} \infty & if\ \gamma_k = 0 \\ \tau_k & \text{otherwise} \end{cases} \tag{13}$$

We have that if we get lost detection at time t we will not expect to receive that detection data in the future.

Thus the measurement model is defined as:

$$y_k = \gamma_k C_k x_k + v_k \tag{14}$$

where $v_k \sim N(0, Q_k)$ is the measurement error covariance. Now we can formulate the delayed output equation 5, following [13], as:

$$y_k^* = \gamma_k C_s^* x_s + v_k^* \tag{15}$$

where $s = k - M$, where M is the number of delayed samples. In comparison to a typical sensor used in robot feedback control, such as a gyroscope, which has a bandwidth of > 500Hz [3], the bandwidth of a DVS sensor depends on several factors as mentioned earlier. In the current work we have analyzed the bandwidth of our experimental set-up, by subjecting the sensor to a pseudo-random amplitude step input, refer to figure 4, to analyze the response and check if the delay is time varying. And in this case we have measured an average delay of $\bar{\tau}_k = -0.71776$ s which equates to a bandwidth of 1.3931 Hz. This delay can play a significant factor in typical robotic systems as the bandwidth of the sensors is relatively low in comparison to the closed loop systems bandwidth.

6 Discussion

It is well known that the performance and robust stability of a control system changes significantly when the plant includes time-delay [23], and as shown earlier, detection loss adds a significant additional delay into the system which can be even larger than τ_k, if no object is detected for long periods of time.

To ensure that the system behaves desirably the following assumption can be made:

Assumption 2. *The closed loop system bandwidth ω_{CL} follows the following inequality $\omega_{CL} < \frac{1}{\max(\tau)}$*

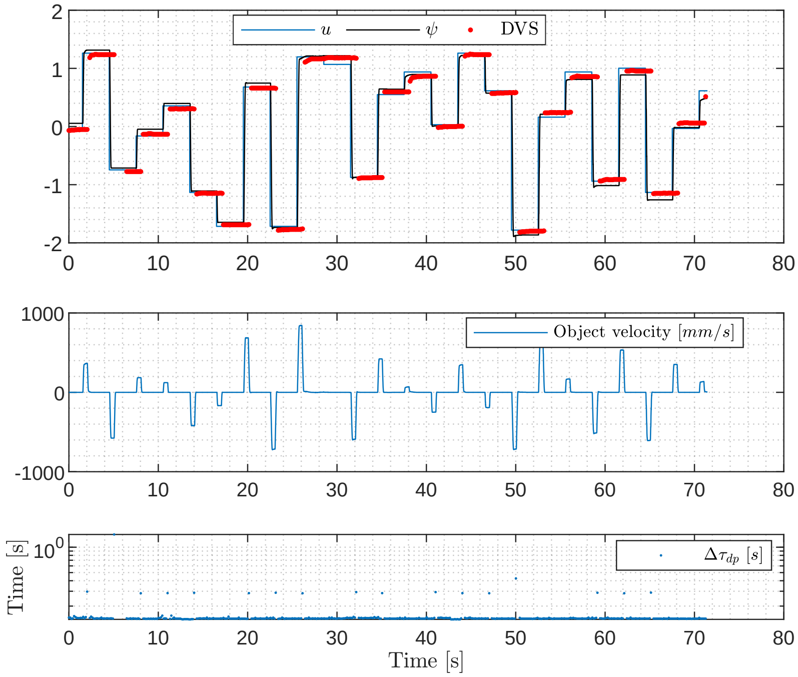

Fig. 4: Experimental Data, the DVS sensor subjected to a pseudo-random amplitude step input.

Due to equation 13, the assumption 2 can be difficult to fulfill, as the system might experience very long delays due to detection loss. This can be solved in a series of ways. One possible solution is to use a low pass filter (LPF), as is commonly used as a way of handling disturbances [23], or by changing the physical properties of the system e.g. the inertia. However, changing the inertia in real time is in most cases not practically feasible. On the other hand a dynamic LPF could easily be implemented, where τ could be estimated using normalized cross-correlation between z and x [1].

Switching Controller Rule

One workaround to manage the detection losses, could be to introduce a hybrid controller structure:

According to assumption 1, we could under certain circumstances have a $\gamma_k = 0,\ for\ k = \{t, \infty\}$, thus we define β as

$$\beta = \begin{cases} 1 & if\ \sum \gamma_k < d_{max} \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

where

$$u = \beta K_1 x + (1 - \beta) K_2 x \tag{17}$$

where $d_{max} \in \mathbb{Z} = \{0, 1, 2, 3, \dots\}$ is the maximal number of detection losses and β is the switching rule. If $\beta = 1$ one controller design option could be to have $K_1 = 0$, i.e. stay still until the DVS sensor has regained tracking on the object. This evidently results in the system reducing its speed and with reduced speed increases the probability of tracking the object.

7 Conclusion

In this work we analysed the effect that the dynamic behaviour of a DVS may have on a control system, through a series of experiments. We then formulated a model for this type of visual servoing system, based on the principles of networked control systems, which models the random delay and detection loss associated with the DVS. One of the key differences between the networked control systems and the DVS is that the detection loss is different in nature to packet loss. The difference being that in the case of a DVS, if detection loss occurs, the data will be lost forever. However, in the case of a networked control system, in most cases the package drop will result in data arriving later in the buffer. In order to use the DVS signal as a feedback to a controller, an estimator would be needed to predict the true state during loss of data. This and how to implement it in our prototype system will be part of our future work. Lastly, we discussed some possible solutions to this problem and some potential control systems issues.

References

1. Finddelay: Estimate delay(s) between signals. https://se.mathworks.com/help/signal/ref/finddelay.htm. Accessed: 2021-24-11.
2. Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.
3. W Geiger, J Bartholomeyczik, U Breng, W Gutmann, M Hafen, E Handrich, M Huber, A Jackle, U Kempfer, H Kopmann, et al. Mems imu for ahrs applications. In *2008 IEEE/ION Position, Location and Navigation Symposium*, pages 225–231. IEEE, 2008.
4. Jake Gemerek, Silvia Ferrari, Brian H. Wang, and Mark E. Campbell. Video-guided camera control for target tracking and following. *IFAC-PapersOnLine*, 51(34):176 – 183, 2019. 2nd IFAC Conference on Cyber-Physical and Human Systems CPHS 2018.
5. Alessandro Giusti, Jérôme Guzzi, Dan C Ciresan, Fang-Lin He, Juan P Rodríguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jürgen Schmidhuber, Gianni Di Caro, et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667, 2016.
6. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
7. Ramyad Hadidi, Jiashen Cao, Yilun Xie, Bahar Asgari, Tushar Krishna, and Hyesoon Kim. Characterizing the deployment of deep neural networks on commercial edge devices. In *2019 IEEE International Symposium on Workload Characterization (IISWC)*, pages 35–48. IEEE, 2019.
8. Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1314–1324, 2019.
9. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.
10. S. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, 1996.
11. Zhen Jia, Arjuna Balasuriya, and Subhash Challa. Vision based data fusion for autonomous vehicles target tracking using interacting multiple dynamic models. *Computer vision and image understanding*, 109(1):1–21, 2008.
12. Danica Kragic, Henrik I Christensen, et al. Survey on visual servoing for manipulation. *Computational Vision and Active Perception Laboratory, Fiskartorpsv*, 15:2002, 2002.
13. Thomas Dall Larsen, Nils A Andersen, Ole Ravn, and Niels Kjølstad Poulsen. Incorporation of time delayed measurements in a discrete-time kalman filter. In *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No. 98CH36171)*, volume 4, pages 3972–3977. IEEE, 1998.
14. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
15. Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

16. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.
17. Antonio Loquercio, Ana I Maqueda, Carlos R del Blanco, and Davide Scaramuzza. Dronet: Learning to fly by driving. *IEEE Robotics and Automation Letters*, 3(2):1088–1095, 2018.
18. Zakariae Machkour, Daniel Ortiz-Arroyo, and Petar Durdevic. Classical and deep learning based visual servoing systems: A survey on state of the art. *Journal of Intelligent and Robotic Systems*, accepted, 2021.
19. Johan Nilsson et al. Real-time control systems with delays. 1998.
20. Durdevic Petar, Daniel Ortiz-Arroyo, Shaobao Li, and Zhenyu Yang. Vision aided navigation of a quad-rotor for autonomous wind-farm inspection. Elsevier, 2019.
21. N. Ramakoti, A. Vinay, and R. K. Jatoth. Particle swarm optimization aided kalman filter for object tracking. In *2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies*, pages 531–533, 2009.
22. Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
23. Emre Sariyildiz, Roberto Oboe, and Kouhei Ohnishi. Disturbance observer-based robust control and its applications: 35th anniversary overview. *IEEE Transactions on Industrial Electronics*, 67(3):2042–2053, 2019.
24. Luca Schenato. Kalman filtering for networked control systems with random delay and packet loss. In *Conference of Mathematical Theroy of Networks and Systems (MTNS'06)*. Citeseer, 2006.
25. Luca Schenato. Optimal estimation in networked control systems subject to random delay and packet drop. *IEEE transactions on automatic control*, 53(5):1311–1317, 2008.
26. Luca Schenato, Bruno Sinopoli, Massimo Franceschetti, Kameshwar Poolla, and S Shankar Sastry. Foundations of control and estimation over lossy networks. *Proceedings of the IEEE*, 95(1):163–187, 2007.
27. Ling Shi, Michael Epstein, and Richard M Murray. Kalman filtering over a packet-dropping network: A probabilistic perspective. *IEEE Transactions on Automatic Control*, 55(3):594–604, 2010.
28. Bruno Sinopoli, Luca Schenato, Massimo Franceschetti, Kameshwar Poolla, Michael I Jordan, and Shankar S Sastry. Kalman filtering with intermittent observations. *IEEE transactions on Automatic Control*, 49(9):1453–1464, 2004.
29. Chi-Yi Tsai, Xavier Dutoit, Kai-Tai Song, Hendrik Van Brussel, and Marnix Nuttin. Robust face tracking control of a mobile robot using self-tuning kalman filter and echo state network. *Asian Journal of Control*, 12(4):488–509, 2010.
30. Wei Zhang, Michael S Branicky, and Stephen M Phillips. Stability of networked control systems. *IEEE control systems magazine*, 21(1):84–99, 2001.