Optimal State Estimation for DNN Visual Servoing Systems with Detection Loss

Novrup, Christian Tranholm; Nowak, Thomas Kjær; Ortiz Arroyo, Daniel; Sahlin, Simon Lennart; Durdevic, Petar

Published in:
2023 11th International Conference on Control, Mechatronics and Automation, ICCMA 2023

DOI (link to publication from Publisher):
10.1109/ICCMA59762.2023.10375025

Publication date:
2023

Document Version
Accepted author manuscript, peer reviewed version

Optimal State Estimation for DNN Visual Servoing Systems with Detection Loss

Christian Tranholm Novrup[a,1], Thomas Kjær Nowak[a,2], Daniel Ortiz Arroyo[a,3], Simon Lennart Sahlin[a,3] and Petar Durdevic[a,3]

***Abstract*—The introduction of deep learning techniques, such as object detection in visual servoing systems, has produced more sophisticated robotic systems capable of working in unknown environments. However, the interaction between the object detection network and the controller, when *detection loss* occurs, has received little attention. In this paper, we investigate a way of mitigating the effect of detection loss in VSNN systems. In our approach detection losses are modeled as a Bernoulli random variable and integrated into the state space model of the dynamic system. To mitigate the effect of detection loss, we propose a variation of a Kalman filter, that artificially inflates the measurement noise covariance when detection loss occurs. The Kalman filter was implemented on a 6DOF robotic manipulator with an eye-in-hand configuration with YOLOv5 as the object detection network. The results show, that the proposed Kalman filter decreases the effects of detection losses and significantly improves performance compared to having a standard Kalman filter, and not having a state estimator at all. The benefit of our approach is especially noticeable when detection loss occurs frequently and for relatively long periods of time.**

***Index Terms*—Visual servoing Detection loss Convolutional Neural Network YOLOv5 Estimation Kalman filter**

I. Introduction

Visual servoing (VS) is a method for controlling the movements of a robotic system using visual input from a camera. As described in [1], the goal of VS in robotic manipulators is to minimize the error between the current pose of the end effector and the desired pose of an object with which the robotic arm is interacting with [2] [3]. Three main types of visual servoing systems have been proposed: 1) image-based visual servoing (IBVS), where the error is based on features extracted from a 2D image, 2) position-based visual servoing (PBVS), where the error is based on parameters in 3D space, and 3) hybrid visual servoing, which is a combination of the two previous methods. There are two commonly used configurations in VS, one where the camera is mounted on the end effector of the robot (eye-in-hand) and one where the camera is fixed in the workspace [1].

In VS applications the camera coupled with image processing algorithms enables a robotic system to detect and identify objects, providing the feedback signal needed to control the movements of the robot.

[a]Authors are with Department of Energy, Aalborg University, 6700 Esbjerg, Denmark
[1]s223145@dtu.dk
[2]thnow22@student.sdu.dk
[3]{doa, sls, pdl}@energy.aau.dk

Classical methods for detecting objects in an image use feature extraction algorithms to identify low level features such as lines and points. However, it is well known that these methods cannot adapt to slightly different variations of objects with similar geometries and to the variety of lighting and background conditions [2], [3] found in unknown environments.

More recently several state-of-the-art methods for visual servoing based on deep neural networks (DNN) have been described in [3]. In these methods DNNs are trained to detect and recognize complex objects in spite of variations on the object size, geometry, orientation or position, and lighting conditions.

An example of a popular object recognition DNN is AlexNet [4]. A derivation of this architecture was retrained in [5] to recognize a specific type of plant leaf that a 6DOF robotic manipulator with an eye-in-hand camera configuration could grab. This research found that convolutional neural networks (CNNs) are capable of generalizing features of objects with a high degree of randomness such as the ones found in leaves to detect/classify these objects. The network in [5] was able to detect the leaves, such that a robotic arm could successfully grab/touch a leaf of a stationary plant, but the system was not capable of handling lost detections produced by the CNN.

Similarly as visual servoing, target tracking systems allow a robotic system to keep a target object within the field of vision of the camera. For instance, in [6], a mobile ground robot with a rigidly mounted camera was developed with the purpose of detecting, tracking, and following a moving human in real-time. The experimental and simulation results showed that the system could successfully track and follow a human using the object detection CNN MobileNet [7]. However, this approach requires the generation of a consistent bounding box for the tracked humans, and bounding box errors are sometimes inferred by the CNN. When this happens, the control system will "blindly" follow the errors, which leads to the robot moving in the wrong direction, eventually losing sight of the human. Furthermore, when object detection data is completely lost, the robot was made to stop at the cost of reducing its tracking performance.

To solve some of the issues that target tracking applications have, Kalman filters are commonly used as state estimators as is reported in [8], [9], [10]. For instance, [8] uses the Interacting Multiple Models (IMM) algorithm to fuse several linear models of a dynamic target object. For

this purpose, an extended Kalman filter is used, whose gain is automatically adjusted. Similarly, in [9], a Kalman filter for state estimation was used in object tracking, but in this case, its noise covariance matrices were fine-tuned through the Particle Swarm Optimization algorithm. In [10] a face-tracking control system of a mobile robot that combines a self-tuning Kalman filter and an echo state network self-tuning algorithm was presented.

CNNs for object detection have clear advantages over traditional feature extraction methods as they tolerate variations in object geometry and illumination conditions [3]. In [11] the term *detection loss* was defined to identify events when no output is produced by an object detection network. This may happen due to several factors, such as the object being too far or too close to the camera, poor lighting, when there are temporary occlusions, or when the robot/target is moving fast. In other research works, detection loss has been referred to as *image loss* [12] [13], but the term *detection loss* describes more accurately the failure of an object detection network to produce an output.

The detection loss effect in VS systems based on CNNs (VSNN) is similar in nature to packet loss, an event that commonly happens in distributed networked control systems (NCS) [11], which was investigated in [14]. The main difference is, that when detection loss occurs in a VSNN, the data is lost forever, whereas in NCS the data may arrive at a later time. To mitigate the effects of packet loss in NCS [14] proposes different types of estimators based on Kalman filters to determine their effect on packet losses and delays. The similarity of packet loss in NCS and detection loss in VSNN suggests looking into applying similar methods for alleviating the effects detection loss. Particularly in [15] a Kalman filter for intermittent measurements is proposed, which makes the filter use an infinite sensor noise covariance when a measurement has not arrived.

Another important problem identified in [11] is the delay produced by the inference processing in the CNN (time between input and output for the CNN), which may be a computationally expensive task when complex DNN are used. The delay issue in VSNN is similar to the delay that happens in NCS due to network congestion. In NCS, delays are commonly handled by using buffers. In the case of VSNN delays can be reduced using faster GPUs, but this will increase the cost of the system. Similarly, as it happens in NCS, when inference delays are introduced into a feedback control system, they may cause system instability [16] since the controller will act on delayed data. This will cause the controller to overcompensate and potentially become unstable. One simple method to compensate for these delays is to design a controller with lower bandwidth, but this is not a desirable solution in many applications.

The effect of detection loss due to target object occlusion in VS is discussed in [12]. In this work, a dynamic model of a robotic system is created to compute missing image data in cases, where a landmark image is momentarily unavailable. The authors use standard feature extraction techniques to detect color, surface, bounding box of the landmark, etc.

The use of Kalman filters for state estimation in VS systems has been investigated in [17]. In this work, an extended Kalman filter (EKF) was utilized in a VS setup for a 5DOF robotic manipulator with an eye-in-hand configuration. The EKF estimated feature points in the image, extracted using classical feature extraction algorithms.

A more recent work [18] uses an EKF on a VS system for a 6DOF manipulator. The main goal of the paper was to estimate, with the help of the EKF, the movement of a target object, they wanted to track. The target was an AprilTag, and feature extraction was done through the ViSP detector. In this work, the low data rate from the camera and the delay due to the feature extraction algorithm were identified as the main bottlenecks of the dynamic VS system. The authors found that the EKF improved the data rate and reduced the effects of delay along with providing an accurate velocity estimate. Moreover [18] showed that an EKF can help to alleviate the effect of delays by providing accurate state estimation.

Despite its relevance, detection loss in VSNN systems has received relatively little attention in the literature [11]. Furthermore, research that has studied methods to handle the loss of visual feedback is mostly based on classical methods for feature extraction that have limited object detection accuracy. Additionally, the few works on VS systems that use DNNs for object detection, implement a simple policy that stops the robot's movement, until object detection data is produced again [6] [5]. Clearly, better methods of handling detection loss and delays are needed to allow a robotic system to continue working despite of these effects, hereby increasing the VSNN system's stability.

This paper investigates the effect of optimal state estimators on VSNN systems for making them capable of handling detection loss and time delays effectively. The main contribution of this paper is:

- A variation of the optimal state estimator Kalman filter is proposed that artificially inflates the measurement covariance when detection loss occurs. This filter is capable of improving the robotic manipulator's performance and stability by reducing the effect that detection loss and delays have on a VSNN.

To our knowledge, no previous research work has studied these issues and used a similar approach to reduce the effects that detection loss and delay have on VSNN-based systems for robotic manipulators.

This paper is organized as follows, section II describes the VSNN system, section III presents a model of our system, section IV discusses state estimators for detection loss, section V briefly describes the controller structure chosen for the experimental work, in section VI, the proposed state estimator is implemented and tested, and section VII and section VIII present our conclusions and future work.

II. System Description

The experimental setup used in this paper is a 6DOF Kinova Gen3 Lite robot manipulator with an RGB-D camera (Intel RealSense L515) [19] mounted on the end effector/gripper (eye in hand). The computing system is a Nvidia

Jetson Nano, controlling the camera via the Kortex API [20] in Python.

For object detection, a pre-trained CNN YOLOv5 (YOLO) [21] was used to detect a bottle. Upon successful detection, YOLO returns the label of the detected object l, the confidence c, top-left pixel coordinates of the bounding box $[x_{min}, y_{min}]$ and the lower-right pixel coordinates of the bounding box $[x_{max}, y_{max}]$. A position error vector between the gripper and the object detected by YOLO was calculated.

First, the bounding box corner coordinates are used to calculate its center coordinates (u, v):

$$u = \frac{x_{max} - x_{min}}{2} \tag{1}$$

$$v = \frac{y_{max} - y_{min}}{2}. \tag{2}$$

With the center of the bounding box as the target for the gripper, the homogeneous vector $\tilde{\boldsymbol{\epsilon}}_{\mathbf{0}}$ (the tilde indicates that the vector is homogenous), which describes the direction between the gripper and the desired object/target, is given by:

$$\tilde{\boldsymbol{\epsilon}}_0 = M^+ \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \tag{3}$$

where M^+ is the pseudoinverse of M, where M is defined as:

$$M = K \begin{bmatrix} R & -R\,\boldsymbol{t} \end{bmatrix}. \tag{4}$$

K is the intrinsic camera matrix [22], R is the rotation matrix, between the gripper and camera reference frame, and t is the translational vector between the gripper origin and the camera origin. $\tilde{\boldsymbol{\epsilon}}_{\mathbf{0}}$ is only a direction vector since it is not possible to reconstruct a 3D point from a 2D image using inverse projection. Therefore a line parametrization is defined from the camera origin in the direction of $\tilde{\boldsymbol{\epsilon}}_{\mathbf{0}}$, where λ is the parameter of the line parametrization:

$$\boldsymbol{\epsilon_b}(\lambda) = \boldsymbol{t} + \lambda(\boldsymbol{\epsilon_0} - \boldsymbol{t}). \tag{5}$$

$\boldsymbol{\epsilon_b}$ is the error vector between the gripper origin and the target, and $\boldsymbol{\epsilon_0}$ is $\tilde{\boldsymbol{\epsilon}}_{\mathbf{0}}$ in Cartesian coordinates. The distance to the target object is measured using the LiDAR. To make sure that the length of the line from the camera origin to the target must be equal to the length measured using the LiDAR we have:

$$d = \left\| \lambda(\boldsymbol{\epsilon_0} - \boldsymbol{t}) \right\|_2. \tag{6}$$

When (6) is solved for λ, and the value of λ is substituted in (5), the error vector $\boldsymbol{\epsilon_b}$ can be obtained. The vector describes the error between the gripper and the object to be manipulated. This error vector is minimized, by moving the manipulator's gripper using visual servoing to reach the target. These relationships, between the different vectors, can be seen in fig. 1. In the figure, O_c, O_b, and O_i denote the origin of the camera's reference frame, the gripper's reference frame, and the image plane, respectively. The black vectors from the origin are used to indicate the basis vectors of that frame. $\boldsymbol{\epsilon_c}$ is the vector to the world point in the reference frame of the camera.

III. Model

The model of the DNN VS system, introduced in [11], is as follows: a discrete-time linear and stochastic plant can be described by (7) [11], [23]:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= A\boldsymbol{x}_k + B\boldsymbol{u}_k + \boldsymbol{w}_k \\ \boldsymbol{y}_k &= \gamma_k(C\boldsymbol{x}_k + \boldsymbol{v}_k). \end{aligned} \tag{7}$$

$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{\epsilon}_b(k+1) & \boldsymbol{\epsilon}_b(k) \end{bmatrix}^T$ is the state vector at sample k, $\boldsymbol{v}_k \sim N(0, R)$ is the sensor noise with covariance matrix R, $\boldsymbol{w}_k \sim N(0, Q)$ is the process noise with covariance matrix Q, and γ_k is a Bernoulli random variable that describes detection loss in the system, and is defined in (8) [11]:

$$\gamma_k^t = \begin{cases} 1 \text{ if detection data arrives before or at time } t, t \leq k. \\ 0 \text{ otherwise.} \end{cases} \tag{8}$$

This essentially means that the output is either zero, or equal to the output equation in the state space model, depending on if a target object was detected by the DNN or not:

$$y_k = \begin{cases} (C\boldsymbol{x}_k + v_k) \text{ if object is detected.} \\ 0 \text{ otherwise.} \end{cases} \tag{9}$$

The A, B and C matrix shown in (7) describes the dynamics of the system/plant, which in case of this paper is explained in the beginning of section II. To find suitable values for these matrices the following basic relationship is assumed:

$$\ddot{\epsilon}_x = -\frac{1}{\tau_x}\dot{\epsilon}_x - \frac{K_x}{\tau_x} v_x, \tag{10}$$

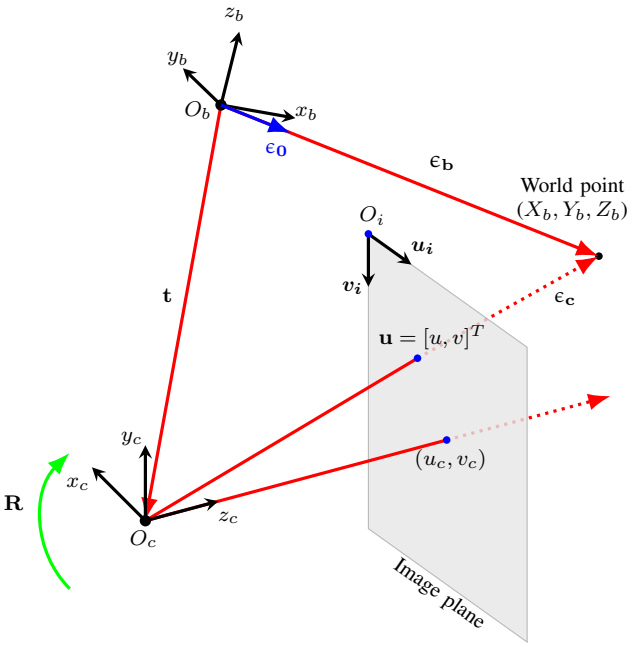


Fig. 1: Visualisation of the camera projection model

which is a basic motion model. ϵ_x indicates the x component of the error vector (position), v_x is the velocity command given to the gripper, and K, τ are model parameters, which can be seen in table I for x, y and z direction (found by identification):

The continuous time model was discretized using the Tustin approximation [16] and a sample time of 0.22 seconds, as it was the fastest sampling time the system could provide.

IV. State Estimation with Detection Loss

To estimate the system's state, a Kalman filter was used together with the model described in section III. The Kalman filter is a modified version of the original Kalman filter, designed to handle detection loss, as was proposed in [15]:

$$\begin{aligned} \hat{\boldsymbol{x}}_{k+1|k} &= A\hat{\boldsymbol{x}}_{k|k} + B\boldsymbol{u}_{k|k} \\ P_{k+1|k} &= AP_{k|k}A' + Q \\ S &= CPC' + R\gamma + (1-\gamma)\sigma^2 I \\ K &= PC'S^{-1} \\ \hat{\boldsymbol{x}}_{k+1|k+1} &= \hat{\boldsymbol{x}}_{k+1|k} + K(y_k - C\hat{\boldsymbol{x}}_{k+1|k}) \\ P_{k+1|k+1} &= (I - KC)P, \end{aligned} \tag{11}$$

where A,B,C, and γ were defined in (7). The filter in (11) will be denoted as Kalman Filter with γ (KF w. γ) for the remainder of the paper. The standard Kalman Filter will be denoted as Kalman Filter without γ (KF w/o. γ). σ is chosen as 10^4 as this was found to be a value where the filter effectively ignores the measurement. From the Kalman filter definition, it can be seen that if $\gamma = 1$ the object is detected and the measurement noise covariance matrix is R. In case $\gamma = 0$ there is no detection and the noise covariance matrix will be $\sigma^2 I$, which will cause the state estimate to reduce trust in the measurement practically entirely. This means, that provided that the covariance matrices R and Q are correct, that the artificially inflated covariance $\sigma^2 I$ is large enough, and that the filter neglects the measurement, the Kalman filter will give the optimal estimate in both modes of operation, i.e. when an object is detected or not. The measurement noise covariance matrix R was experimentally determined to be:

$$R = 10^{-6} \begin{bmatrix} 0.1009 & -0.0198 & -0.0220 \\ -0.0198 & 0.1913 & 0.0028 \\ -0.0220 & 0.0028 & 0.2913 \end{bmatrix} \tag{12}$$

V. Control

In (11) and (7), the input to the system is given by

$$\boldsymbol{u}_{k|k} = -K_p \, \hat{\boldsymbol{x}}_{k|k}, \tag{13}$$

where K_p is a matrix of proportional gains, such that the controller feeds back the error vector $\boldsymbol{\epsilon}_b(k)$, and converts these to a velocity input for the gripper. The values in K_p were ad-hoc tuned.

TABLE I: Model parameters

	x	y	z
K	1.22	1.013	1.176
τ	0.3	0.16	0.16

VI. Closed Loop Experiment with the Estimator

To test the system including the estimator and controller, an experiment was set up with an object to detect right in front of the manipulator. The gripper was given an initial starting position below the object and an initial constant velocity of 0.05 m/s in the x-direction was given, such that it would sweep past the object. The control objective was then to move the gripper in the y-direction, such that it would center the object in the camera frame. The experiment is shown in fig. 2 and a video can be found in [24].

In the top left of fig. 2 the manipulator is in its initial position with the gripper far to the right and below the object, which is not yet in view. In the second stage, on the top right, the manipulator has moved slightly to the left and has detected the target object (a bottle). The gripper will now start moving up in the y-direction to make the center of the bottle align with the red line on the camera view. In the third stage, bottom left, the gripper has succeeded in centering the red line on the camera with the center of the bottle, so the camera and bottle are at the same height. The gripper is still moving to the left with a constant velocity. In the final stage of the experiment, in the bottom right, the bottle is still centered on the y-axis of the image but is now far to the right of the manipulator. At this stage, the manipulator cannot move further to the left, so the experiment ends. In all the experiments detection losses were simulated in software, as the successful detections and their losses were not consistent. To produce comparable, consistent, and replicable results, the detection losses were induced periodically according to (14)

$$\gamma_a(t) = \begin{cases} 1 \; nT \leq t \leq nT + DT, \; n = \{0, 1, 2, ...\} \\ 0 \text{ else} \end{cases} \tag{14}$$

where γ_a denotes the artificial detection loss (analogous to γ in (11) but not a random variable), D is the detection loss ratio between 0-1 (determines detection loss duration), T is the switching period, and t is time. n is incremented every time $t = nT + DT$. $\gamma_a(t)$ will be a square wave with duty ratio D. Inducing detection loss artificially and periodically is not an accurate representation of the real detection losses that occur randomly, where the system may never regain detecting of target objects after it is lost. However, this approach was

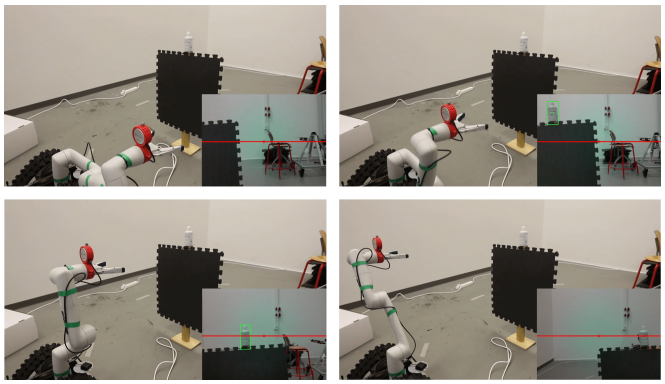


Fig. 2: Four stages of the experiment

assessed to be suitable for investigating the effectiveness of the filter, compared to the case of KF w/o. γ and not using a KF.

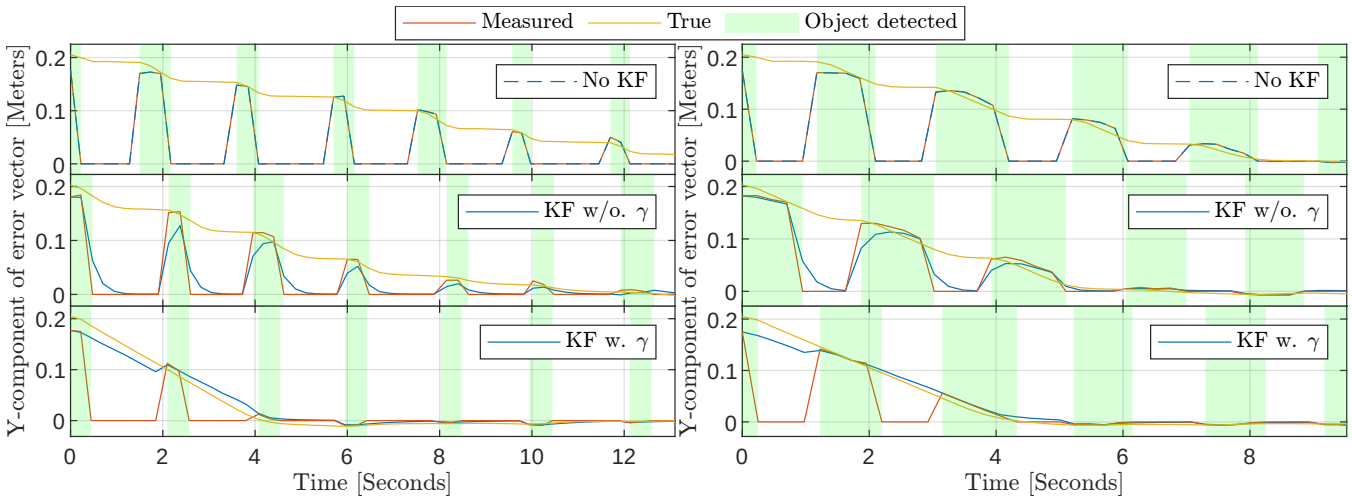
Two tests were carried out with different detection loss ratios (0.25 and 0.5) and the results can be seen in fig. 3.

The P controller adjusted the y-position (vertical) of the manipulator to place the object in the middle of the frame, minimizing the y component of the error vector. At both detection loss ratios, three different experiments were carried out. One experiment without the KF, then with a KF w/o. γ, and finally with a KF w. γ.

Looking at the experiment with a detection loss ratio of 0.25 (fig. 3a), it can be seen, that without KF, the y-position of the gripper does not reach a steady state within the experiment duration. The gripper only moves in the y-direction whenever a detection is successful (green background), and it stops the y-direction movement whenever there is no successful detection. The KF w/o. γ experiment shows, that the controller reaches a steady state within the experiment duration after around 11 seconds. The y-position of the gripper moves the fastest when the detection is successful and slows down when a measurement is lost. This is because when a detection is lost, the estimate of the y-position of the gripper from the KF w/o. γ goes to zero. Comparing the system without KF and KF w/o. γ, the gripper moves a little during detection loss for KF w/o γ, which causes a faster settling time than the system without KF. The KF w. γ moves regardless if detection occurs or not, which makes the y-position of the gripper settle at around 4 seconds, making it the fastest of the experiments. When the detection is successful, at around 2 seconds, the system corrects the estimated position (blue line) to meet the measured value (red line). In the experiment with a detection loss ratio of 0.5 (fig. 3b), the detections are successful for twice as long as the previous experiment. It can also be seen, that in the 0.5 detection loss experiments then the system settles considerably faster than the experiment with a detection loss ratio of 0.25 for the No KF and KF w/o γ. This is because most of the control action in the system happens when detection is successful. This means, that the settling times of these methods are dependent on the time when detections are successful. With the KF w. γ however, the settling time is very similar for both detection loss rates, showing that the KF w. γ decreases the effects of detection loss effectively. It can also be seen, that the KF w/o γ is oscillating between the measured signal and zero, the reason for this comes from (9). The KF w/o γ has a constant measurement noise covariance matrix, meaning it has no way of differentiating a successful detection, versus when detection loss happens, which will always return a zero.

VII. Discussion

The proposed Kalman filter was shown to reduce the effects of detection loss significantly in comparison to a standard Kalman filter and to the system without Kalman filter as is summarized in table II. This table shows a decrease in the settling time t_s by using KF w. γ, of roughly 50% for both detection loss rates, compared to not using a KF. Comparing KF w. and w/o γ, a decreased settling time of about 40% (0.25) and 23% (0.5) can be achieved using KF w. γ depending on the detection loss rate. The results show that the filter is effective, especially at higher detection loss rates, where there are only few detections. These experimental results furthermore match results and expectations from simulations of the proposed Kalman filter. In these experiments, the detection loss was artificially induced. This allowed showing, that the filter performs well even in the case of having a small number of successful detections. Similar behavior is expected when the filter is tested in real conditions. The filter used a fixed noise covariance, that was



(a) Detection loss of 0.25

(b) Detection loss of 0.50

Fig. 3: Test of the estimator, with the manipulator moving with a constant velocity of 0.05 m/s in the x- direction, and a P-controller adjusting the y- position.

artificially inflated during detection loss. However, there are other ways to do this to get a more accurate noise covariance during detection loss and to avoid filter divergence due to the use of incorrect noise covariances [25].

TABLE II: Settling time for true movements of fig. 3 being within ± 0.01 meter

	Detection loss ratio 0.25		Detection loss ratio 0.5	
	t_s [s]	Decrease [%]	t_s [s]	Decrease [%]
No KF	14.15	0	7.9	0
KF w/o. γ	10.73	24.17	5.32	32.66
KF w. γ	6.42	54.63	4.08	48.35

VIII. Conclusion

Detection loss is the event that happens when an object detection DNN misses detecting an object due to several causes. This phenomenon has thus far received little attention. In this paper the effect that detection loss has on VSNN systems was investigated. We propose a variation of a Kalman filter, that artificially inflates the measurement covariance, when detection loss occurs, causing the filter to disregard the measurement and estimate the states of the system based on the linear system model. The method was tested on a visual servoing system consisting of a Kinova 6DOF manipulator with an Intel RealSense LiDAR camera in an eye-in-hand configuration. The experiment showed, that the proposed Kalman filter decreases the effect that detection loss has on the controller to achieve a significantly faster settling time, compared to the use of a conventional Kalman filter and to a system without Kalman filter. In our experiments, detection losses were induced periodically to compare the results of the different system configurations, but in the real world, detection loss occurs randomly. The control system used was an ad-hoc tuned P-controller and the measurement covariance matrix was a static matrix, artificially inflated during detection loss. For future work, we plan to test our system with real random detection losses, implement an adaptive control to handle the delay from the detection loss, and perform online covariance estimation.

References

[1] S. Hutchinson, G. Hager, P. Corke, A tutorial on visual servo control, IEEE transactions on robotics and automation 12 (5) (1996) 651–670.

[2] D. Kragic, H. Christensen, F. A, Survey on visual servoing for manipulation, Comput. Vis. Act. Percept. Lab. Fiskartorpsv 15 (02 2002).

[3] Z. Machkour, D. Ortiz Arroyo, P. Durdevic, Classical and deep learning based visual servoing systems: A survey on state of the art, Journal of Intelligent and Robotic Systems 104 (1) (Jan. 2022).

[4] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2017) 84–90.

[5] K. Ahlin, B. Joffe, A.-P. Hu, G. McMurray, N. Sadegh, Autonomous leaf picking using deep learning and visual-servoing, IFAC-PapersOnLine 49 (16) (2016) 177–183, 5th IFAC Conference on Sensing, Control and Automation Technologies for Agriculture AGRICONTROL 2016.

[6] J. Gemerek, S. Ferrari, B. H. Wang, M. E. Campbell, Video-guided camera control for target tracking and following, IFAC-PapersOnLine 51 (34) (2019) 176–183, 2nd IFAC Conference on Cyber-Physical and Human Systems CPHS 2018.

[7] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications (2017).

[8] Z. Jia, A. Balasuriya, S. Challa, Vision based data fusion for autonomous vehicles target tracking using interacting multiple dynamic models, Computer vision and image understanding 109 (1) (2008) 1–21.

[9] N. Ramakoti, A. Vinay, R. K. Jatoth, Particle swarm optimization aided kalman filter for object tracking, in: 2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies, 2009, pp. 531–533.

[10] C.-Y. Tsai, X. Dutoit, K.-T. Song, H. Van Brussel, M. Nuttin, Robust face tracking control of a mobile robot using self-tuning kalman filter and echo state network, Asian Journal of Control 12 (4) (2010) 488–509.

[11] P. Durdevic, D. Ortiz Arroyo, Dynamic analysis and modeling of dnn-based visual servoing systems, in: Dynamic Analysis and Modeling of DNN-based Visual Servoing Systems, Springer, Germany, 2022.

[12] D. Folio, V. Cadenat, A sensor-based controller able to treat total image loss and to guarantee non-collision during a vision-based navigation task, in: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2008, pp. 3052–3057.

[13] Z. Zhang, T. M. Bieze, J. Dequidt, A. Kruszewski, C. Duriez, Visual servoing control of soft robots based on finite element model, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 2895–2901.

[14] L. Schenato, Kalman filtering for networked control systems with random delay and packet loss (01 2006).

[15] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. Jordan, S. Sastry, Kalman filtering with intermittent observations, IEEE TRANSACTIONS ON AUTOMATIC CONTROL 49 (9) (2004) 1453–1464.

[16] G. F. Franklin, J. D. Powell, A. Emami-Naeini, Feedback control systems, 8th edition, Pearson, NA.

[17] W. Wilson, C. Williams Hulls, G. Bell, Relative end-effector control using cartesian position based visual servoing, IEEE transactions on robotics and automation 12 (5) (1996) 684–696.

[18] A. Oliva, E. Aertbeliën, J. de Schutter, P. Robuffo Giordano, F. Chaumette, Towards Dynamic Visual Servoing for Interaction Control and Moving Targets, in: ICRA 2022 - IEEE International Conference on Robotics and Automation, IEEE, Philadelphia, United States, 2022, pp. 1–7.

[19] Intel, Intel® realsense™ lidar camera l515 datasheet, https://docs.rs-online.com/f31c/A700000006942953.pdf, (Accessed on 1/02/2023).

[20] Kinovarobotics/kortex: Code examples and api documentation for kinova® kortex™ robotic arms, https://github.com/Kinovarobotics/kortex, (Accessed on 06/07/2022).

[21] Yolov5, https://github.com/ultralytics/yolov5, (Accessed on 06/07/2022).

[22] M. Sonka, Image processing, analysis, and machine vision., 3rd Edition, Cengage Learning, Stamford CT, 2008.

[23] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, S. S. Sastry, Foundations of control and estimation over lossy networks, Proceedings of the IEEE 95 (1) (2007) 163–187.

[24] Results 2 - youtube, https://www.youtube.com/watch?v=0ymSimncsm8, (Accessed on 01/02/2023) (2022).

[25] R. Moghe, M. R. Akella, R. Zanetti, Riemannian trust-region based adaptive kalman filter with unknown noise covariance matrices (2021).