Real-time Visual Tracking for Aided Inertial Estimation of Wing Shape

Aditya Pradeep Menon[1] and Umer Naeem Khan[2]
L'Institut Supérieur de l'Aéronautique et de l'Espace

***Abstract*— Amongst many challenges, de-carbonisation presents as one of the main strategies employed by the aviation industry, to tackle climate change. This issue has forced the industry to take drastic steps to reduce carbon footprint and consequentially mitigate the effects of climate change. One of the ways to concur with the future needs is lowering fuel consumption and hence, lowering emissions, which may come at the price of higher operating costs. This could be achieved by introducing flexible aircrafts which have large wingspans, and therefore less induced drag. These result in more structural flexibility but introduce the potential for control instabilities. To address these limitations, active structural control and stability augmentation systems must be implemented. Therefore, the in-flight estimation of the wing shape of such aircraft is essential to build these structural and stability control laws. This paper proposes a methodology in estimating flexible structural states based on a Computer Vision technique, exploiting the use of a camera, on a simple flexible wing prototype. This provides an initial shape estimation solution. Extended Kalman Filtering is then employed with the aid of additional low-bandwidth sensors, to bound diverging estimation errors. This proposed technique of Kalman filtering coupled with Computer Vision, allows for an accurate real-time estimation of large nonlinear wing deflections. The technique is verified by means of real-time simulations of a deflected flexible wing prototype, using a tracking algorithm that estimates the position of LEDs placed all along the wing, and using a Kalman filter model that works in tandem with the tracking algorithm.**

I. INTRODUCTION

Whilst keeping in mind the needs for the aviation industry and satisfying strict power requirements in solar powered high-altitude pseudo-satellite aircraft (HAPS), there is an increasing trend to optimise aerodynamic performance by increasing aspect ratio and to reduce weight by minimising structure. The HALion project aims to create such aircraft, and these come under the category of High Altitude Long Endurance (abbreviated HALE) aircrafts. The various structural modes interact with autopilot laws, requiring the need for estimation in real-time of the aircraft structure. Such flexible systems require protective control laws for managing load distribution during gusts and assure dynamic structural stability [1].
There are various techniques that have been explored by researcher in this domain, and these are explored in this section.

[1]A. Menon is currently enrolled in a Masters of Science in Aerospace Engineering, majoring in Embedded Systems at ISAE-SUPAERO.
[2]U.Khan is currently enrolled in a Masters of Science in Aerospace Engineering, majoring in Embedded Systems at ISAE-SUPAERO.

The vision-based system proposed by Sanches *et al.* [2] provides evidence that computer vision can be used to estimate wing deformation. The aircraft X-HALE, is a highly flexible aircraft and the proposed system measures this wing's deflections.

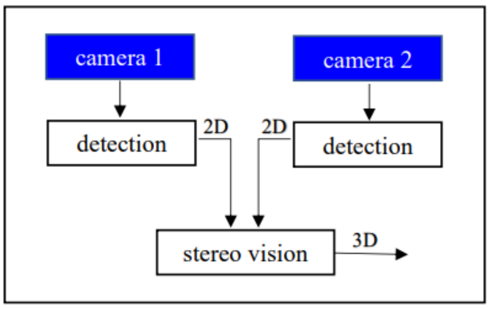

Fig. 1: Workflow Diagram of Methodology, [2]

Figure 1 presents the methodology approach used by Sanches *et al.* [2]. The vision-based system acquires data through a pair of cameras installed on the wing of the X-HALE. Red markers have been placed on the wing which serve as reference points to represent the wing deformation. Two computer vision strategies are implemented, namely the red object detection procedure and the stereo vision procedure. The former detects 2D coordinates of red markers using a color detection algorithm, while the latter uses triangulation to obtain 3D coordinates.

Experimental results demonstrate that this methodology successfully detects red markers on the wing and estimates the wing shape offline. It is successfully able to obtain physical coordinates by simply using a set of two cameras and a color detection algorithm. Therefore, this study indicates that computer vision techniques can be applied for estimating wing shape and deformation, providing potential for control and improvement of highly flexible aircraft.[2]

To the best of the author's knowledge, this paper is the first attempt to use use real-time visual tracking for aided inertial estimation of wing shape estimation. Visual tracking is performed using Computer Vision techniques. Whilst existing applications have already coupled Computer Vision (henceforth referred to as CV) and Extended Kalman Filtering (henceforth referred to as EKF) to bound estimation errors, the proposed technique focuses on real-time implementation and hardware compatibility. By allowing for the reconstruction of wing shape in real time, the method offers a promising solution for improving wing shape estimation in practical settings.

II. ARCHITECTURE OF THE SYSTEM

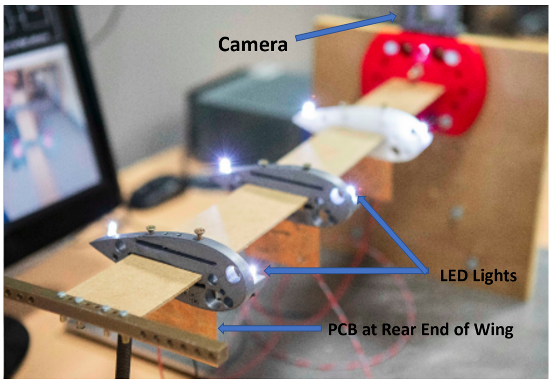

Fig. 2: Prototype of the Model

Figure 2 is the prototype of the model that has been used for the entirety of this project. This experimental high aspect ratio wing has been constructed with a square rod handle at the tip which can be used for placing weights to simulate different scenarios. The wing has three NACA0018 airfoils placed along the wing, each having a constant chord length of 14mm. Each airfoil comes equipped with a Printed Circuit Board (henceforth referred to as PCB) and two Light Emitting Didoes (henceforth referred to as LED) which have been used as markers to implement CV techniques. The camera has also been placed at the root of the wing such that it can be readjusted if needed to capture all the markers. It should be noted that each PCB is fitted with an Inertial Measurement Unit (henceforth referred to as IMU).

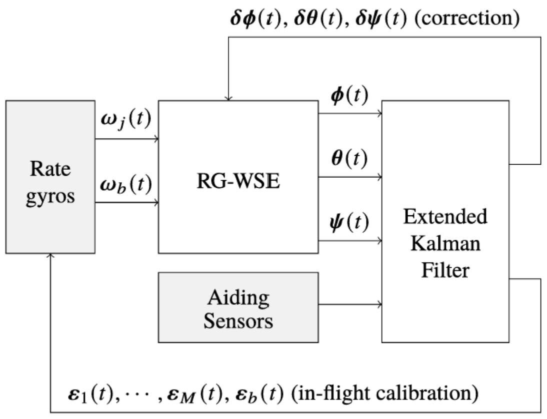
δφ(t), δθ(t), δψ(t) (correction)
Rate gyros
ω_j(t)
ω_b(t)
RG-WSE
φ(t)
θ(t)
ψ(t)
Extended Kalman Filter
Aiding Sensors
ε_1(t), ··· , ε_M(t), ε_b(t) (in-flight calibration)

Fig. 3: Aided Inertial Wing Shape Estimator Overall Architecture [1]

Figure 3 illustrates the overall architecture of the proposed method for estimating the wing shape of an aircraft during flight. The IMUs are responsible for capturing the wing's orientation data. The IMU provides this orientation data to the EKF, which uses aeroelastic modeling and wing shape parameterization to estimate the wing shape. The estimated wing shape is then refined through shape optimization, which adjusts the parameters of the wing shape until the estimated wing shape closely matches the actual wing shape. [1]

The authors have decided to implement CV techniques to bound the estimation errors. Essentially, this is a part of the block titled ”Aiding Sensors” in Figure 3. A model was designed on MATLAB [1] where synthetic values were generated to get an insight into how Computer Vision can optimise the results. The model encapsulates all concepts which are relevant to the project at hand. The IMU data from the prototype and the stream from the camera is fed directly to the the model to do the same. Certain sections of the model have been modified to fit the needs of the project such that the need for synthetic data is eliminated. The model helped the authors to proceed with vision based wing shape estimation.

III. Vision-based Wing Shape Estimation

This section describes the technique and methodology used for the Real-time Visual Tracking for Aided Inertial Estimation of Wing Shape. The block of ”Aiding Sensors” will be modified based on the data given by a Computer Vision tracking algorithm, adapted by the authors.

A. *Tracking over Detection*

Tracking in simplest terms is locating an object in successive frames of a video. In Computer Vision and Machine Learning applications, tracking is considered to be a broad term which encompasses many concepts. For this project, the emphasis has been placed on multiple-object tracking. In multiple-object trackers, the first frame of each object is marked using a rectangular box, to indicate the location of the object that is to be tracked. The objects are then tracked in subsequent frames using a particular tracking algorithm. The authors have chosen tracking over detection after taking into account several factors. The following list gives the reader a brief insight into the same:

1. Tracking learns about the object using previous frames.
2. Under occlusion, tracking can prevail over detection. This concept will be demonstrated later in the report with the use of the actual tracking algorithms.
3. Tracking preserves identity of the desired target.

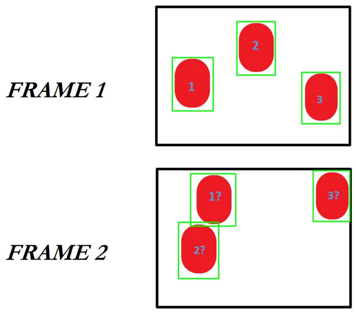

Fig. 4: Frame by Frame Object Detection

As can be seen in Figure 4, the three objects that are being detected are red dots which are identical to one another. In the first frame, the detection algorithm correctly identifies the three objects by placing a rectangular box around them. In the next frame, the algorithm succeeds in detecting the three dots again which is a positive sign. However, it is not able to make the link between each of these dots in the two consecutive frames. This implies that it is not able to tell which dot represents "Dot 1" in the subsequent frame. In the context of this project, this poses a grave problem since the objects that are being tracked are identical LED lights. Therefore, object tracking seems to be a more viable choice with respect to this project's objectives. Before delving deeper into the project, it is important for the reader to have a good understanding of certain concepts from a practical standpoint.

The objective in tracking is to find the desired object in the current frame, given that it's been successfully tracked in previous frames. It does so through two models: appearance and motion model. The object in the first frame is selected using a bounding box and up until the current frame, the object has been moving. This means that its motion model is now determined. Motion model means that the algorithm knows the location and the velocity of the object in previous frames. Based on this current motion model, it is easier to predict where the object will be in the next and subsequent frames. Furthermore, the appearance of the object is also being learned continuously. So, the motion model and the appearance model work in tandem to provide an accurate estimate of where the object is.

The appearance model works on the principle of a classifier that is trained in an online manner'[3]. The classifier, as the name suggests, classifies a rectangular region of an image as either the object that is to be tracked or the background. The way the classifier knows if the object is within this rectangular region is by scoring this image patch 0 or 1. If the score is 1, the classifier knows with 100% certainty that the patch contains the object and if the score is 0, it is sure that it is the background.

This classifier is trained in an "online" manner, referring to the training of the algorithms that occurs in real-time. An online classifier's particular benefit is that one can get it up and running with only a few examples provided to the algorithm in run-time, whilst for an offline classifier, it is necessary to train it with many examples.

As can been seen in Figure 5, a classifier is trained by feeding it many positive (contains the object) and negative (background) examples. In this example, the classifier is initialized with many positive samples where the face of the woman is being detected using various bounding boxes. Then, the classifier is trained using negative samples where some of the face is occluded or the detection is not fully covered. Once this is initialized with thousands of examples, the algorithm running the classifier is executed. One can notice that using these samples, the algorithm is able to locate the object. If the object isn't located, the classifier is trained with even more samples (in the case where the object is not detected, it classifies it as the background) and updates in a loop, while moving on to the next frame. With an online classifier, the algorithm will update the tracking process in an "online" fashion, meaning the classifier is updated using input image frames. This paper will review algorithms that use an online classifier.

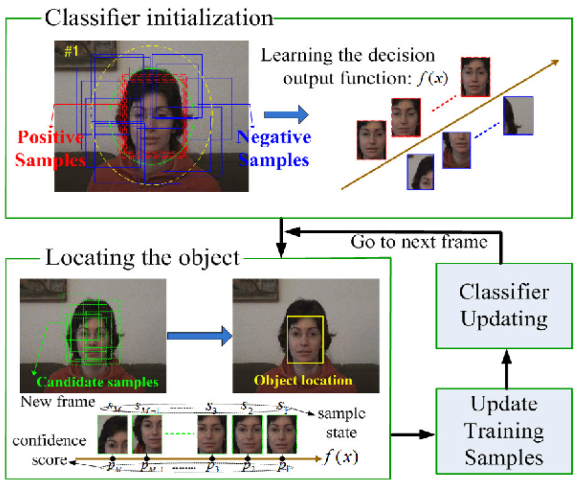

Fig. 5: Classifier Model [4]

B. Tracking Algorithm

As vision techniques are reliant on cameras, camera calibration is necessary, to ensure tracking algorithms take into account the internal and external parameters of the camera model. This is important in tracking purposes, to account for any possible lens distortions. The calibration method used is Zhang's chessboard Method, wher the objective is t find the intrinsic camera properties and distortion coefficients.[5]. Having performed the calibration technique on the Raspberry PI fish-eye Camera, Table I shows the intrinsic camera matrix coefficients and Table II shows the distortion coefficients.

TABLE I: Intrinsic Camera Matrix

Camera Matrix		
637.497	0	312.455
0	850.543	253.129
0	0	1

TABLE II: Distortion Coefficients

Distortion Coefficients				
0.0813	0.822	0.0129	0.00264	-5.467

Having introduced the concept of classifiers and the general overview of tracking algorithms, this section introduces the tracking algorithm that is used to estimate the shape of the wing in real-time. It is important to note the objects that will be tracked, using the CV algorithm, are the LEDs placed on the wing. The choice of using LEDs to track is primarily because the LEDs are a well-lighted object and are placed all along the wing, allowing to measure the deflection at each point.

It is important to note that the tracking algorithm must perform under circumstances such as the following: Occlusion, Object Out-of-View, Illumination changes, and Agile Motion. Out-of-view is a particular challenge as, if the LEDs move out of the view of the cameras, the visual tracking estimation will fail and thus, the tracking algorithm must recover and continue tracking once the LED is back in frame. Therefore, the process of choosing a tracking algorithm is how it performs under these situations.

Furthermore, the computational tracking time is a necessary performance parameter. The tracking time can either be evaluated by the frames per second (henceforth referred to as FPS) required for the tracker to robustly track the LED, or by the computational time taken for the tracker to complete its estimation. In the context of object tracking, FPS defines how fast the object tracking model processes the video feed and generates the desired outputs.

Taking these qualitative and quantitative performance criteria into account, Table IV, shown in Section V displays the results obtained after testing 8 OpenCV tracking algorithms.

It is to be noted that the qualitative performance criteria is purely judged by the eye. The user executing the command to track the objects can judge how well the tracking algorithm performs under such criteria shown in Table IV.These results were taken for multiple object tracking, that is the 6 LEDs placed on the wing. Kernelized Correlation Filter (henceforth referred to as KCF) & the Channel Spatial Reliability Tracker (henceforth referred to as CSRT) perform the best under all situations. KCF's computational tracking time is lower than CSRT's, and its FPS generated is greater, therefore making it a viable choice for the LED tracking algorithm. It is important for the tracking algorithm to minimize the delay that would be required by the EKF to estimate the wing shape. Therefore, it is necessary for the tracking algorithm to output its results with the least amount of delay.

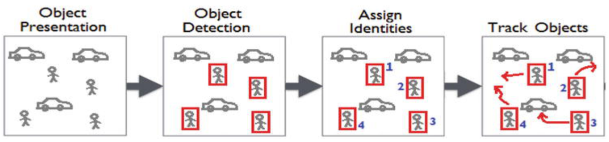

Fig. 6: Tracking Algorithm Overview [6]

A typical scenario of visual tracking is to track an unknown object initialized by a bounding box in subsequent image frames. The tracker is initialized with its input parameters, the region of interest is selected (bounded by the video frame), and the tracking of the object is performed. Figure 6 shows the general overview of the tracking algorithm. It should be noted that this photo is purely for understanding the high-level architecture of the algorithm. For the HALion project, the frame is presented first, the LEDs are detected, each LED has a unique ID, and the tracking is performed. The purpose of this type of tracker is obtain the position of the current target object from the previous position and to separate the discriminative background and object. This class of tracking technique is called "tracking by detection". They come under the category of discriminative model trackers.

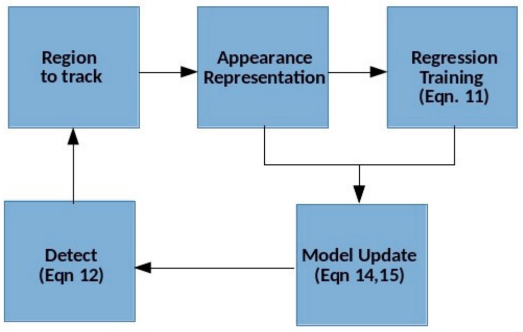

Fig. 7: KCF Tracking Algorithm [7]

Figure 7 shows the architecture of how the KCF tracking algorithm functions. KCF is a type of discriminative model tracker, that capitalizes on Correlation Filters. Although correlation filters require a large number of samples to "learn" about the object, hence introducing a computational burden, KCF takes advantage of circulant structure of a data matrix with higher samples, and expands it using a kernel trick, to increase the computational speed of the tracker [8].

KCF defines a target size that it takes from the bounding box's location (the position coordinates and the size of the frame). It then uses bilinear interpolation for image resizing, and HOG features are used in the tracking algorithm. The extracted patch and each RGB channel are weighted by a cosine window before a Fourier transformation. The extraction of the HOG features from patches is then followed by an implementation of the Gaussian correlation on the 2D Fourier transform of the features to identify the maximum response, which is the target location. The process is repeated for each new frame, and the model is updated by training it on the new patch at the new location. Finally, the updated position is saved and used for detection in the next frame.

C. *Accelerating tracking performance*

As mentioned previously, it is necessary for the tracking algorithm to compute as fast as possible, providing a minimum delay in the EKF control loop. Therefore, it is possible to take advantage of a graphics processing unit (henceforth referred to as GPU) computing power, to accomplish this task. The wing prototype contains a mini-computer, an NVIDIA Jetson Nano. The NVIDIA Jetson Nano module uses a GPU unit of 128 cores, much higher than the ordinary laptop, and an ARM Cortex-A57, a quad core CPU [9]. To capitalize on this GPU unit, the tracking algorithm was adapted using certain CUDA C++ functions. Cuda C++ is a programming language extension that enables developers to leverage the power of NVIDIA GPUs for parallel computing. By offloading computationally-intensive tasks to the GPU, CUDA C++ can significantly accelerate the tracking process and improve the overall performance of CV applications. In the context of object tracking, CUDA C++ can be used to perform multiple tasks in parallel, such as object detection, feature extraction, and motion analysis, leading to faster processing times and increased frame rates.

TABLE III: CPU and GPU Performance Comparison

KCF Algorithm	**CPU Performance**	**GPU Performance**
Computational Time	54 sec	5 sec
FPS	50 fps	650 fps

Table III shows the comparison between tracking using just CPU power compared to performing the tracking intensive part of the algorithm by the GPU. It can be seen that there is a significant improvement in the computational time and the FPS generated. The FPS generated by using GPU is roughly 13 times better and the computational time is also significantly lower.

IV. Results & Analysis

This section describes and analyses the results obtained, from the tracking algorithm. Table V shows each algorithm's performance qualitatively and quantitatively. It was concluded that the KCF tracker seemed to be the most performant tracker, with faster computational tracking time and higher FPS output. The algorithm's efficiency can be attributed to several factors:

- Kernel trick: The KCF algorithm uses a kernel function to map the features of the target object into a high-dimensional space, where the target and its appearance can be tracked more easily. This allows the algorithm to efficiently compare the features of the target object with those in the candidate regions without explicitly computing the high-dimensional feature vectors, thus saving a lot of computation time.
- Circulant matrix: The KCF algorithm uses a circulant matrix to model the correlation between the target object and its appearance in the search image. This matrix can be efficiently computed using the Fast Fourier transform (FFT), which makes the algorithm much faster than other correlation-based trackers that use the full matrix.
- Adaptive learning: The KCF algorithm uses an adaptive learning strategy to continuously update the model as the object moves and changes appearance. This ensures that the tracker remains accurate and robust, even when dealing with challenging scenarios like occlusion or out-of-view situations.
- Parallel processing: The KCF algorithm can be easily parallelized, allowing it to take advantage of multi-core processors and GPUs to further speed up the tracking process (e.g. using Cuda C++ functions).

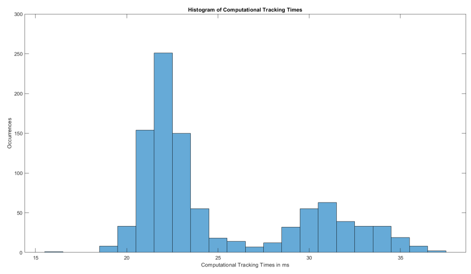

Fig. 8: Computational Tracking Times

Figure 8 shows a histogram that displays the computational tracking times for the KCF tracking algorithm and the respective number of occurrences. These computational tracking times are the result of a timer that calculates the amount of time it takes for the tracker to perform tracking of the LED from one frame to the next frame, until the end of the video feed. One can notice that the most frequent tracking time is between 21.5 and 22.5 milliseconds.

For the real-time system to be bounded, the computational tracking times should be within a certain range or bound, that does not exceed 22.5 milliseconds. It can be seen though, that there are some frames in the video, where the tracking algorithm takes more computational time to track the LED. Therefore, the algorithm should be optimized to ensure that the real-time constraint is met and a low latency data is achieved, so that a minimum delay is introduced in the control loop of the flexible aircraft wing shape estimation.

V. Conclusions

This paper proposed a novel technique for real-time visual tracking for the aided inertial estimation of wing shape.

Real-time Visual tracking is an important technique when it comes to tracking sensors placed on the wing of a very flexible aircraft. Specifically, tracking of LED is performed as an aiding sensor, in estimating the deflections a wing undergoes when subjected to aerodynamic forces introduced by a flexible aircraft wing. This allows for an additional technique used for the aided inertial estimation of wing shape, coupling it with the EKF, which helps to bound diverging estimation errors arising from the EKF. The EKF model did not contain any computer vision technique before, and with this proposed technique, it completes the aiding sensors block in the EKF model. The proposed technique inherits the benefits of Extended Kalman filtering, as well as multi-rate asynchronous aiding sensors integration. This study demonstrates that integrating the camera system tightly leads to a complete state estimation using only one camera, which differs from earlier research that employed stereo vision techniques (requiring at least two sighting devices). However, there is still a lot of scope for future research, such as producing a fully integrated model, that includes the camera sensor data, as well as the IMUs data, coupled with the EKF estimation. Further, visual tracking data can be validated, by comparing it with data obtained from a motion capture room that detects the deflections the wing undergoes.

Appendix

TABLE IV: Tracking Algorithms Performance

	Quantitative Performance Criteria		**Qualitative Performance Criteria**			
Algorithm	Computational Tracking Time	FPS	Occlusion	Out-of-View	Agile Motion	Illumination Changes
BOOSTING	103 sec	15 fps	NO	NO	NO	NO
MIL	224 sec	4 fps	NO	YES	NO	NO
KCF	54 sec	50 fps	YES	YES	YES	YES
TLD	215 sec	6 fps	NO	NO	YES	NO
MEDIANFLOW	54 sec	50 fps	YES	NO	YES	NO
GOTURN	10 sec	150 fps	NO	NO	NO	NO
MOSSE	40 sec	300 fps	NO	NO	YES	NO
CSRT	132 sec	10 fps	YES	YES	YES	YES

Acknowledgment

This material is based upon work supported by Doctor Leandro Lustosa, Director of ION**LAB** and Associate Professor of Aeronautics and Aerospace at ISAE-SUPAERO. The work is also supported by Airbus in the frame of the Airbus-Michigan Center for Aero-Servo-Elasticity of Very Flexible Aircraft (CASE-VFA).

References

[1] L. R. Lustosa, I. Kolmanovsky, C. E. Cesnik, and F. Vetrano, "Aided inertial estimation of wing shape," Journal of Guidance, Control, and Dynamics, vol. 44, no. 2, pp. 210–219, 2021.

[2] A. C. Sanches, F. J. Silvestre, M. A. V. Morales, and H. Hesse, "Designing a vision-based system to measure in-flight wing deformation," Proceedings of the 31st Congress of the (ICAS), pp. 1–7, 2018.

[3] S. Mallick, Object tracking using opencv (c++/python), en, [online], Available at: 20 June 2022, 2017. [Online]. Available: https://learnopencv.com/object-tracking-using-opencv-cpp-python/.

[4] C. Deng, Y. Han, and B. Zhao, "High-performance visual tracking with extreme learning machine framework," IEEE Transactions on Cybernetics, vol. PP, Dec. 2018. DOI: 10.1109/TCYB.2018.2886580.

[5] W. Burger, "Zhang's camera calibration algorithm: In-depth tutorial and implementation," p. 53, May 2016. DOI: 10.13140/RG.2.1.1166.1688/1.

[6] D. Shah, Multi object tracking, https://medium.com/visionwizard/object-tracking-675d7a33e687, [Accessed 27-Mar-2023], 2020.

[7] M. George, B. R. Jose, and J. Mathew, "Performance evaluation of kcf based trackers using vot dataset," Procedia Computer Science, vol. 125, pp. 560–567, 2018.

[8] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," IEEE Transactions on Pattern Analysis; Machine Intelligence, vol. 37, no. 03, pp. 583–596, Mar. 2015. DOI: 10.1109/TPAMI.2014.2345390.

[9] Sheshadri and Franklin, Introducing the nvidia jetson nano 2gb developer kit, https://developer.nvidia.com/embedded/jetson-nano-2gb-developer-kit, 2020.