Adaptive real-time clustering method for dynamic visual tracking of very flexible wings

Mkhoyan, Tigran; de Visser, Coen C.; De Breuker, Roeland

DOI
10.2514/1.I010860

Publication date
2021

Document Version
Final published version

Published in
Journal of Aerospace Information Systems

JOURNAL OF AEROSPACE INFORMATION SYSTEMS
Vol. 18, No. 2, February 2021

Adaptive Real-Time Clustering Method for Dynamic Visual Tracking of Very Flexible Wings

Tigran Mkhoyan,* Coen C. de Visser,† and Roeland De Breuker‡
Delft University of Technology, 2600 GB Delft, The Netherlands

https://doi.org/10.2514/1.I010860

Advancements in aircraft controller design, paired with increasingly flexible aircraft concepts, create the need for the development of novel (smart) adaptive sensing methods suitable for aeroelastic state estimation. A potentially universal and noninvasive approach is visual tracking. However, many tracking methods require manual selection of initial marker locations at the start of a tracking sequence. This study aims to address the gap by investigating a robust machine learning approach for unsupervised automatic labeling of visual markers. The method uses fast DBSCAN and adaptive image segmentation pipeline with hue-saturation-value color filter to extract and label the marker centers under the presence of marker failure. In a comparative study, the DBSCAN clustering performance is assessed against an alternative clustering method, the disjoint-set data structure. The segmentation-clustering pipeline with DBSCAN is capable of running real-time at 250 FPS on a single camera image sequence with a resolution of 1088×600 pixels. To increase robustness against noise, a novel formulation (the inverse DBSCAN, DBSCAN^{-1}) is introduced. This approach is validated on an experimental dataset collected from camera observations of a flexible wing undergoing gust excitations in a wind tunnel, demonstrating an excellent match with the ground truth obtained with a laser vibrometer measurement system.

Nomenclature

A, B	=	subsets of dataset D
$B(x', y')$	=	kernel matrix
$\bar{\boldsymbol{c}}_p$	=	centroid of cluster centers
$\bar{\boldsymbol{c}}_{\text{cp}}$	=	centroid of points $\boldsymbol{P}$
D	=	dataset
$\text{dist}(p, q)_{\text{euclid}}$	=	Euclidean distance function
$f(I(x, y))$	=	filtering (sequence) operation
$f_{\text{dilate}}(I(x, y))$	=	dilate operation
$f_{\text{erode}}(I(x, y))$	=	erode operation
f_g	=	gust vane frequency
$f_{\text{morph}}(I(x, y))$	=	morphological operations (combined)
f_{norm}	=	global normalization operation
$G_f(x, y)$	=	filtered image
$I(x, y)$	=	input image
$I(z)$	=	Gaussian noise probability density
$J(x, y)$	=	noise input image
MaxPts	=	DBSCAN^{-1} max points dense region
MinPts	=	DBSCAN min points dense region
m_i	=	cluster center size
$N(x, y, t)$	=	random seed initialized noise mask
$N_\epsilon(p)$	=	ϵ neighborhood of points p
n_i	=	cluster population size
$n_{i_{\text{noise}}}$	=	noise particle population size
$O(\dots)$	=	computational complexity
$P(x, y)$	=	density distribution of particles (2D)
$\boldsymbol{P}(x, y)$	=	cloud of cluster centers
$\boldsymbol{P}_{\theta\text{hull}}$	=	convex radial hull
p, q	=	scatter points
p_n, q_n	=	scatter noise particles
$R_{yy}(\tau)$	=	autocorrelation function
$S_{yy}(\omega)$	=	autopower spectral density
V_∞	=	wind tunnel flow velocity
w_1, w_2	=	class variance weights (Otsu)
$Z_\epsilon(p_n)$	=	ϵ neighborhood of noise points p_n
z	=	grayscale value
α_g	=	gust vane angle
γ	=	radius tolerance
ϵ	=	radius of neighboring points
$\boldsymbol{\theta}_{\text{cp}}$	=	vector angles from centroid to marker
μ	=	mean of $I(z)$ distribution
μ_{dst}	=	mean of the cluster population
μ_I	=	mean of points in 2D image
σ	=	standard deviation of $I(z)$ distribution
σ_{dst}	=	standard deviation of cluster population
σ_I	=	standard deviation points in 2D image
$\sigma_w^2(\tau_{\text{th}})$	=	intraclass variance (Otsu)
σ_1^2, σ_2^2	=	class variances (Otsu)
τ_{th}	=	threshold parameter

Presented as Paper 2020-2250 at the AIAA SciTech 2020 Forum, Orlando, FL, January 6–10, 2020; received 17 May 2020; revision received 18 September 2020; accepted for publication 8 November 2020; published online 20 January 2021.

*Ph.D. Student, Faculty of Aerospace Engineering, Aerospace Structures and Materials Department, P.O. Box 5058; t.mkhoyan@tudelft.nl.

†Assistant Professor, Faculty of Aerospace Engineering, Control and Operations Department, P.O. Box 5058; c.c.devisser@tudelft.nl.

‡Associate Professor, Faculty of Aerospace Engineering, Aerospace Structures & Computational Mechanics, P.O. Box 5058; r.debreuker@tudelft.nl.

I. Introduction

IN THE context of aeroservoelastic control, monitoring the entire wingspan can be crucial for proper delegation of control actions. This objective may involve installing many conventional accelerometers that are likely subject to noise and bias, must deal with certification requirements, or might face challenges associated with correct geometric placement or limited mounting space. A smart sensing approach is desired for those examples of wing structures that rely on novel types of sensors for providing feedback to an intelligent controller.

A solution that can significantly reduce the complexity associated with hardware installation and provide the flexibility needed for employing novel state estimation techniques is aeroelastic state estimation by visual methods. A schematic of aeroelastic state estimation using vision (consisting of an intelligent controller, the aircraft model, and the visual model) is illustrated in Fig. 1. This study aims to contribute to the aeroelastic state estimation block, such that the control loop can be closed with the dotted line.

The use of visual information for observing deformations has been successfully implemented on wind tunnel models in early studies [1], and has also seen widespread application in robot manipulation [2]. However, in recent years, the capability in terms of onboard computation and camera quality has immensely increased, whereas the

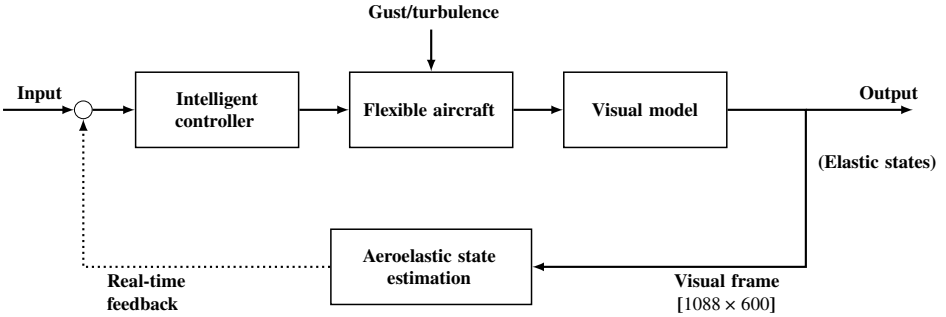


Fig. 1 State estimation setup using visual tracking.

hardware has become more compact [3,4]. These developments open the door for numerous embedded applications using a camera as a sensor for aircraft systems.

Various vision applications for areal systems have been investigated for areal imagery in recent studies [5–7]. Vision-based information can also be used for aerial navigation [8] and flight control tasks, such as aerial refueling [9], landing [10], and estimation of rigid-body aircraft states, such as altitude [11]. However, within the scope of flexible and morphing aircraft systems (as these systems are more prone to exhibiting higher responses to aerodynamic loads), the challenge lies in estimating the impact of the flexibility on the dynamics of the system, which cannot always be accounted for in the early design stage. The study by Weisshaar [12] highlights the ability to monitor and communicate structural state information as one of the key aspects of the smart morphing structures development. Vision-based feedback systems can play a crucial role in this task as one camera system can observe multiple nodes of the system's flexible states in a sequence of images [13]. In particular, fuselage-mounted camera systems can provide significant advantages for flexible aircraft systems, save costs associated with installation and certification, and have the potential of being noninvasive and universally applicable. Vision-based information has been shown to be suitable for direct real-time feedback of flexible states of an aircraft [14,15].

Image data are also a rich source of information: data collected over an extended period of time unlocks the opportunity to approach the state estimation from a new perspective using machine learning methods. One of the key challenges is the need for robust, unsupervised, and computationally efficient clustering methods. Several studies investigated the performance of clustering methods by using improved [16,17] and parallel DBSCAN methods [18]. However, a gap remains for a streamlined approach to unsupervised clustering with robustness against noise. In particular, although many suitable tracking methods exist for marker detection, correctly labeling the initial markers in the visual frame is still not a trivial task [19].

In this study, two machine learning methods were implemented for unsupervised clustering of marker labels, meaning that they do not require the number of clusters and initial guesses as input. The sequence of images is filtered with two image segmentation approaches to obtain a mask for clustering operations. A comparison was made between the two machine learning methods, DBSCAN [20] and disjoint-set data structure [21], and a segmentation-clustering pipeline was developed based on hue-saturation-value (HSV) [22] and adaptive thresholding with Otsu's method [23].

A novel approach to DBSCAN (the inverse DBSCAN, DBSCAN^{-1}) was introduced and implemented in the study. In this approach, the clustering problem is reformulated into a noise filtering problem, and an additional parameter, MaxPts, is introduced into the formulation. The crux of DBSCAN^{-1} lies in isolating the group of desired clusters and classifying them as *noise*, i.e., points surrounded by *too many other points* (filtered by max MaxPts condition). Subsequently, the desired clusters of points are rejected as noise, whereas the *true noise* in the data is identified explicitly and removed from the dataset in a follow-up step.

For the purpose of investigating the robustness of the method, the input images were subjected to Gaussian noise, and both the nominal DBSCAN as well as DBSCAN^{-1} were assessed in performance with

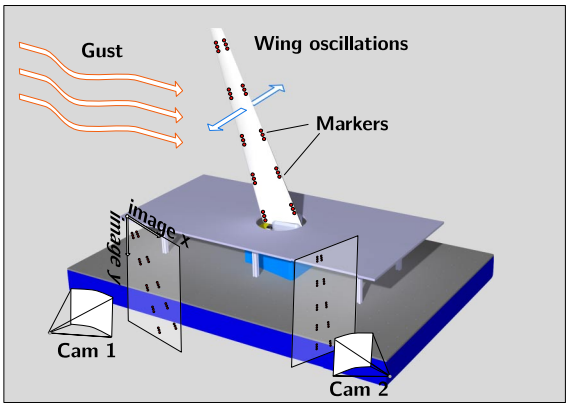


Fig. 2 Experimental setup with the wing facing the wind tunnel, equipped with visual markers.

less noise filtering. An image tracking pipeline was developed to test this clustering method on an image sequence. It was observed that the proposed method is capable of real-time tracking and achieving speeds of 250+ frames per second (FPS), measured on an image sequence of a single camera with a resolution of 1088 × 600 pixels in a laboratory environment on a standard Dell Optiplex 7400 and a 2.3 GHz Intel Core i5 16G MacBook. Hence, the method is suitable for online control applications.

The approach was tested on an image sequence of a flexible wing equipped with light-emitting diode (LED) markers, undergoing oscillatory motion under gust excitation in the Open Jet Facility (OJF) wind tunnel of the Delft University of Technology. Furthermore, the effect of the frequency content was studied to investigate a potential implementation in the pipeline for adjusting the segmentation and clustering parameters. A schematic of the experimental setup is shown in Fig. 2; in this experiment, the same gust generator was used as the one developed for OJF in a previous study [24].

This paper is structured as follows. The methodology is presented in Sec. II, where Sec. II.B deals with the segmentation and filtering approach. Two clustering methods, DBSCAN and disjoint-set data structure, are discussed in Sec. II.C, with a detailed description of the novel formulation of the DBSCAN, DBSCAN^{-1}, in Sec. II.D. The experimental setup and the data acquisition process are explained in Sec. III, with Secs. III.A–III.C covering the setup, hardware, and experimental conditions. Furthermore, Sec. III.D expands on how the validation dataset was created by the automatic labeling tool specifically designed for this study, and Sec. III.E covers the performance test developed for a comparative assessment of the two earlier presented clustering methods. The results of the clustering methods and the full tracking pipeline deployed on the experimental data are discussed in Sec. IV. Finally, the conclusions and recommendations are presented in Sec. V.

II. Methodology

The method proposed in this study describes a computer vision and machine learning approach composed of a robust segmentation-clustering pipeline that is capable of automatically detecting and

extracting marker locations and dealing with temporary marker loss. An image filtering pipeline (segmentation) is implemented consisting of HSV filter and the Otsu's automatic thresholding method [23]. Two machine learning routines are then evaluated: (clustering) DBSCAN [20] and disjoint-set data structure [21]. The segmentation pipeline is used to extract the point data of the markers and the clustering is used to correctly label the cluster centroids. The approach was tested on an image sequence of a flexible wing undergoing motion, equipped with active LED markers.

A. Overview of the Full Tracking Pipeline

A high-level overview of the full tracking pipeline developed for this study is shown in Fig. 3. The segmentation block refers to the segmentation and HSV filtering processes, addressed in Sec. II.B. The red block is the clustering algorithm (DBSCAN/DBSCAN^{-1}/disjoint) implemented in this study as detailed in Secs. II.C and II.D. The green blocks represent an independent tracking filter and Kalman filter duos (KCF-EKF) that run in parallel to keep track of the markers through a sequence of images. The output is the displacement of the marker in (x, y) pixel coordinates of the frame. The cyan block is an additional sorting step needed for consistent tracking of the markers, explained in Sec. II.E. The algorithms presented in this study are mainly concerned with the dotted part as shown in the schematics in Fig. 3 and aim to highlight the methodology needed to arrive at the inverse DBSCAN (DBSCAN^{-1}) algorithm, the main contribution of this study.

B. Segmentation

Segmentation approaches are generally focused on finding a filter or a sequence of filters $f(I(x, y))$ in order to shape an input image $I(x, y)$ to the desired output $G_f(x, y)$ by altering the pixel intensity values:

$$G_f(x,y) = f(I(x,y)) \quad I(x,y) \longrightarrow \boxed{f} \longrightarrow G_f(x,y)$$

For a sequence of images, the process is a function of the number of frames and thus, implicitly, time [25]. When the desired segments of the image contain color information, a commonly applied technique is color filtering in the HSV space. The main benefit of processing in this color space is that the image intensity and color can be distinctly separated. The method also has wide use in video sequence processing and image extraction [22,26].

1. *HSV Filter*

To separate the background from the markers, an HSV filtering pipeline composed of multiple filters is used. First, the image is segmented based on the color temperature of distinct LED markers, based on distinct values of hue, saturation, and value. The filter is tuned to find the near-optimal HSV values to minimize the noise in the image. In Fig. 4, the result is shown of such an operation.

The figures, from left to right, show how the original image is filtered based on its HSV values, obtaining a binary black-and-white (BW) color-filtered image. Then, default thresholding is applied to remove the scattered noise from the light diffusion from LEDs and the remaining background. The result is a BW image, a binary mask with distinct LEDs. Hereafter, contours of the shapes contained in the binary mask are extracted and the clustering can be applied to identify individual markers. The contours extraction filter is based on the Topological Structural Analysis algorithm of binary images and shapes [27], where a border-following technique is applied with the aid of topological analysis of the contours of a border shape.

In Fig. 4, the HSV operation is shown when the images are tracked in low lighting conditions. When lighting conditions change, HSV filtering operation may produce a noisy mask, meaning that aside from a distinct mask with LEDs, additional scattered background pixels are present in the HSV (middle) image. Because this image is close to bimodal by nature, it was investigated how the bimodal Otsu's thresholding can improve the segmentation with an additional HSV filtering step based on the image histogram. In Fig. 5, a simplified schematic is shown of the HSV segmentation and clustering pipeline.

2. *Morphological Operations*

The HSV filter alone may produce a noisy speckle masked image. A typical way to deal with this is by means of morphological image transformations [28]. Morphological operations are, in general, useful, not only for removal of global noise (e.g., Gaussian noise), but also for isolating and joining separate individual elements.

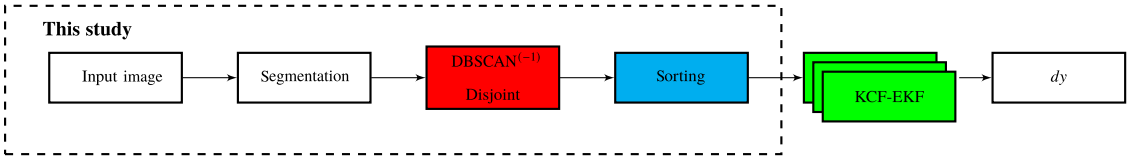


Fig. 3 High-level overview of the full tracking pipeline.

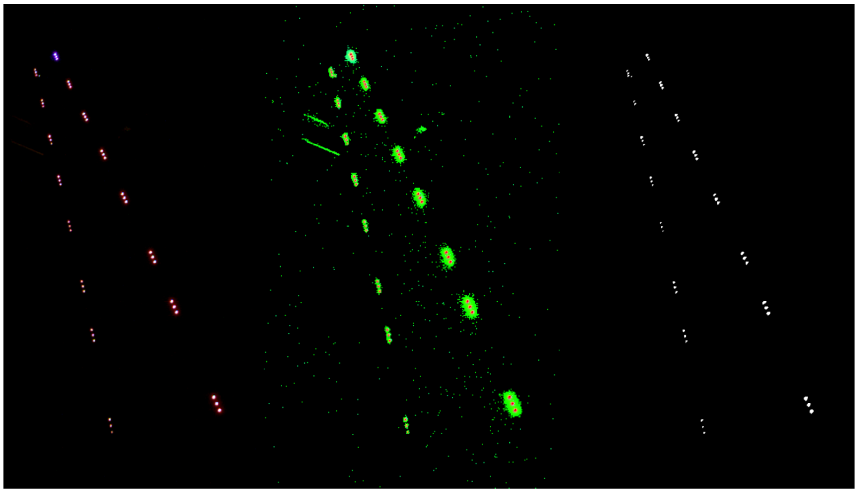


Fig. 4 Single HSV filtering operation: original (left), HSV (center), and black-and-white (BW) threshold (right) image.

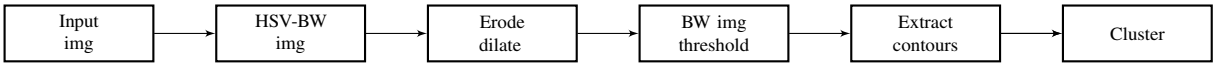


Fig. 5 Schematic of HSV filtering and thresholding resulting in BW image needed for the subsequent clustering process.

A commonly used cascaded operation is *erode*, followed by *dilate*, where the former *erodes* away pixels and pixel groups captured by a certain kernel size, and the latter *dilates* and enlarges bright pixel groups. Both of these image transformations perform, in essence, a convolution operation of image $I(x, y)$ with kernel $B(x', y')$. Erode operator performs a local min operation with a kernel of desired size (e.g., 3×3), anchored at the center. As the kernel slides over the image, the pixel value under the anchor point is replaced by the min value of the region covered by the kernel $B(x', y')$. Dilate operator works according to the same principle, but performs a local max operation. The operations can be summarized as follows:

$$f_{\text{erode}}(I(x, y)) = \min_{(x',y')\in B_{\text{ker}}} (I(x + x', y + y')) \tag{1}$$

$$f_{\text{dilate}}(I(x, y)) = \max_{(x',y')\in B_{\text{ker}}} (I(x + x', y + y')) \tag{2}$$

and combined operation:

$$f_{\text{morph}}(I(x, y)) = f_{\text{dilate}}(f_{\text{erode}}(I(x, y))) \tag{3}$$

For an appropriate kernel size, this will remove away noisy speckles surrounding and scattered around thresholded shapes. In this study the kernel size was set to 2×2 pixels. The relevance and effect of morphological operations will be further discussed in Sec. IV.

3. Thresholding

The thresholding strategy in image processing is essential for obtaining a good mask for DBSCAN clustering. Variations of light and motion activity of the object make the task of obtaining good thresholding for live images challenging [29]. A robust approach has to anticipate the variations in the pixel intensities to produce the best possible mask. Several methods are possible; in this study, three approaches are investigated: global unit normalization, baseline normalization, and adaptive global thresholding using Otsu's method [23].

a. Global Normalization Thresholding. The global normalization can be applied by converting the three-channel RGB input image to grayscale. Subsequently, the image can be scaled with the maximum value of the grayscale, depending on how the grayscale is represented: (0, 1) or (0, 255). Then, a single threshold can be applied to obtain a binary mask $G(x, y)$. For an input image $I(x, y)$, this process can be represented as

$$G(x, y) = f_{\text{norm}}(I(x, y)) \tag{4}$$

$$G(x, y) = \begin{cases} 1, & I(x, y)_{\text{norm}} \geq \tau_{\text{th}} \\ 0, & I(x, y)_{\text{norm}} < \tau_{\text{th}} \end{cases} \tag{5}$$

where the $I(x, y)_{\text{norm}}$ can be computed using a simple scaling, or mean μ_I and standard deviation σ_I of the image:

$$I(x, y)_{\text{norm}} = \frac{I(x, y) - \mu_I}{\sigma_I} \tag{6}$$

The downside of this approach is that it does not take into account the variations in pixel intensities throughout the image sequence that may have been influenced by changing light conditions and/or movement of the object being tracked. The threshold parameter τ_{th} is, in this case, obtained and tailored for a single static image. The quality of the thresholding then depends on the carefully chosen threshold parameter and predictability of the light variations. When applied correctly to a continuous image sequence, in this particular application, an arbitrary thresholding routine should be able to segment the foreground as moving object (high intensity) and detect background as static (low intensity).

b. Baseline Thresholding. In this approach, the baseline pixel intensities are taken into account of the kth image. The first image is a good basis to obtain a suitable threshold parameter such that variations are taken into account from these baseline values. This process can be represented in a way similar to Eq. (5), but now the normalization of the kth sequential image (in range $i = 1, 2, \ldots, N$) is done according to

$$I(x, y)_{\text{norm}_{i=k}} = \frac{I(x, y)_{i=k} - \mu_{I_{i=k}}}{\sigma_{I_{i=k}}} \frac{1}{I(x, y)_{i=0}} \tag{7}$$

The downside of this approach is that the sensitivity to the threshold parameter increases, and the intensities lie closer together. However, an offset is maintained concerning the baseline in each image sequence.

c. Adaptive Otsu Thresholding. Otsu's method is an automatic global thresholding method that tries to categorize an image in two classes, background and foreground pixels [23,30,31]. The method is well suited for images that have a bimodal gray pixel intensity histogram; in this case, the histogram will show two distinct peaks and sharp separation between them, where one peak is assumed to correspond to the bins of the background and the other to the foreground. The threshold value is chosen such that the interclass variance is minimized, which would suggest placing the threshold value in the middle of the peaks. The minimization procedure for finding a threshold value of τ_{th} can be represented as

$$\sigma_w^2(\tau_{\text{th}}) = w_1(\tau_{\text{th}})\sigma_1^2(\tau_{\text{th}}) + w_2(\tau_{\text{th}})\sigma_2^2(\tau_{\text{th}}) \tag{8}$$

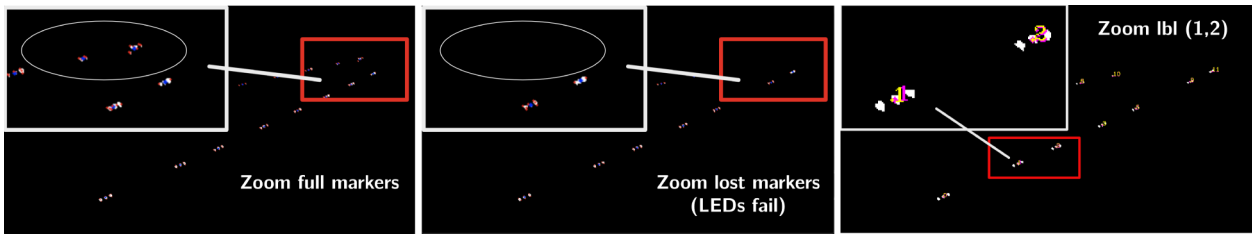
where the parameters w_1, w_2 and σ_1^2, σ_2^2 correspond to the probability and the variance of the two classes and can be computed from the histograms [23].

The limitation of this method is the bimodality assumption, which may not hold for each image and its grayscale image pair [32]. When the object is considerably smaller than the surrounding background, the histogram may not show clear distinctions. Additionally, noise may affect the histogram representation. Variations of Otsu's algorithm exist that are capable of dealing with noisy images [30]; however, in this regard HSV filtering is responsible for filtering out most of the image noise, making the thresholding less complicated.

C. Clustering Approach

To tackle the problem of correctly detecting and clustering the markers, a machine learning approach is used. This study implements and compares two machine learning methods for clustering, DBSCAN [20] and the disjoint-set data structure [21]. These algorithms were particularly suitable due to their unsupervised nature, namely, 1) minimum needed domain knowledge, 2) ability to find clusters of varying size, and 3) ability to deal with noise (in case of DBSCAN). DBSCAN differs from the disjoint-set data structure by its ability to deal with noise in the dataset and achieves the goal at a significantly lower computational cost ($\mathcal{O}(n \log(n))$). The two unsupervised clustering algorithms are implemented in the marker recognition pipeline and are evaluated for performance in terms of speed and robustness.

In this study, it was crucial to apply a robust unsupervised clustering method such that an arbitrary number of markers could be accounted for automatically. The robustness assessment was implemented in the experimental conditions, where, due to failure of the LEDs (going on and off), the number of markers (and thus cluster centers) varied over time and across experimental runs from a nominal (complete) marker set. Within a single experimental run, the failure was mainly of periodic nature and manifested itself due to



a) Complete set of markers **b) Successful clustering of an incomplete set of markers** **c) Comparison DBSCAN scan (purple), and the disjoint-set data structure (yellow)**

Fig. 6 The necessity of unsupervised clustering: incomplete (right, middle) versus full (left) set of markers.

high gust loads and wing oscillations. The clustering assessment of an incomplete and complete set of markers is illustrated in Fig. 6. Here, the red dots are contours of point groups found in the clustering mask, and the blue dots are their respective centroids. The image on the right shows the result of clustering.

1. DBSCAN

The main principle of DBSCAN is to identify and separate regions of high-density from low-density regions. At any given point p, density is measured within a circular radius ϵ. A dense region of radius ϵ from point p is a region that contains at least a MinPts number of points; MinPts and ϵ are the main parameters of the algorithm. Given a database D, the ϵ neighborhood N_ϵ of point p w.r.t. point q has the following form [20]:

$$N_\epsilon(p) = \{q \in D | \text{dist}(p, q) \leq \epsilon\} \tag{9}$$

This definition alone, when used naively, will fail to distinguish core points (points inside the cluster), border points (points at the border of a cluster), and noise (a point not belonging to any cluster). The reason is that, generally, the ϵ neighborhood of border points has much fewer points than the ϵ neighborhood of a core point. The problem arises when the MinPts parameter is set to a low value to include the border points, which can cause noise to be included in the cluster as well. To overcome this DBSCAN introduces the concept of density reachability. A point is said to be *directly density reachable* when the following two conditions hold:

$$p \in N_\epsilon(q) \tag{10}$$

$$|N_\epsilon(p)| \geq \text{MinPts} \quad \text{(core point condition)} \tag{11}$$

These conditions, thus, set a requirement for every point p in a cluster to be in the ϵ neighborhood of another point q in this cluster. Additionally, the ϵ neighborhood of q, $N_\epsilon(q)$, must have a minimum of MinPts, classifying it as a core point. The method further introduces connectivity conditions for connecting N_ϵ of points and defines noise as a point not belonging to any cluster in dataset D under the given conditions (density-reachability and connectivity) [20]. The basis of the clustering approach and the definitions are illustrated in Fig. 7. As shown, point p can be reachable from point q by density-reachability or connectivity.

2. Disjoint-Set Data Structure

Disjoint-set data structure operates by organizing a set of elements into a distinct number of disjoint sets, also referred to as equivalence classes [21]. For a given data set D, obtained as a result of filtering and contour operations, equivalence classes are defined that are non-overlapping. Subsets A and B are considered a disjoint-set when the overlap U between them belongs to an empty set $\varnothing$:

$$A \cap B = \varnothing \tag{12}$$

The algorithm assigns *all* points of the dataset to an equivalence class, hence no inherent mechanism is built-in to cope with noise, and a noise particle may belong to a dedicated subset C. Consequently, and as will become more evident in the following sections, a good filtering approach is needed with this method to remove the noise.

To make the method comparable to DBSCAN, the threshold for the disjoint sets can be defined with a distance metric, radius γ, similar to ϵ. A set of points $\{p, q, \ldots\}$ belongs to a disjoint-set A, when they are packed within radius tolerance γ, resulting in the following conditions:

$$A = \{p, q, \ldots\} \tag{13}$$

$$\text{dist}(p, q)_{\text{euclid}} \leq \gamma \tag{14}$$

Here, the latter condition is defined as the Euclidean norm of points p and q:

$$\text{dist}(p, q)_{\text{euclid}} = \sqrt{(p(x, y) - q(x, y))^2} \tag{15}$$

An illustration of the presented definitions is provided in Fig. 8. In Fig. 6c, a comparison is shown of the clustering operations for DBSCAN scan (purple) and disjoint-set data structure (yellow).

D. Inverse DBSCAN: DBSCAN^{-1}, a Novel Clustering Approach for Sparse Datasets

Although DBSCAN allows explicit definition for noise in the data (points not meeting the core points condition), the success in rejecting the noise is closely tied to the correct selection of parameters and the quality of the thresholded input image. The clustering becomes harder when high-density noise is introduced into the data. Noise can have various sources, e.g., interference in hardware signal, poor illumination, or, simply, poor prefiltering and thresholding of the input image. There are also conditions where prefiltering, such as the

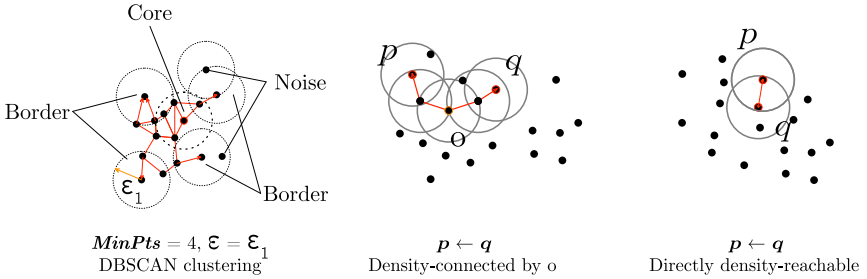


Fig. 7 Illustration of the DBSCAN clustering method.

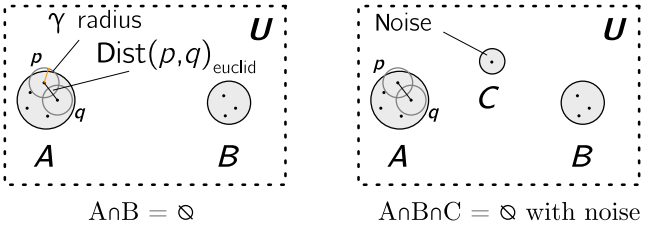


Fig. 8 Venn diagram and illustration of disjoint-set clustering method.

morphological operations, are not possible or have adverse effects (further elaboration follows in Sec. IV). In particular for sparse datasets, under such conditions, DBSCAN is known to fail to identify the desired clusters [33]. This shortcoming arises from the fact that for a high density of scattered noise, noise particles are more likely to meet the core point criteria for a given DBSCAN parameter set.

To remedy this problem, a novel formulation of DBSCAN is proposed, the inverse DBSCAN, denoted by DBSCAN^{-1}. In this new model, a different perspective on the clustering problem is needed: instead of trying to *reject* the noise, it is proposed to *actively look* for noise. Hence, DBSCAN^{-1} tries to explicitly detect noise, and clustering becomes an implicit task. The proposed approach would be to use this formulation of DBSCAN as a noise removal filter, then apply nominal DBSCAN again on the clean image domain. To enable this approach, redefinition of DBSCAN is needed. For a given database D, the ϵ neighborhood of noise particles p_n and q_n is defined as

$$Z_\epsilon(p_n) = \{q_n \in D | \text{dist}(p_n, q_n) \leq \epsilon\} \tag{16}$$

DBSCAN in its original form was intended for obtaining clusters for large datasets and relatively low noise, and hence no limitation is set on the maximum number of clusters. In the definition of DBSCAN^{-1} an additional parameter, denoted by MaxPts, is introduced, which sets a cap on the allowable number of points in the ϵ neighborhood of noise p_n, denoted by Z_ϵ. The noise particle is directly reachable from another cluster of noise particle(s) when the following holds:

$$p_n \in Z_\epsilon(q_n) \tag{17}$$

$$\text{MaxPts} \geq |Z_\epsilon(p_n)| \geq \text{MinPts} \quad \text{(corenoise particle condition)} \tag{18}$$

Three conditions must be placed on the DBSCAN^{-1}: 1) MinPts must be set to 1 to capture individual noise particles; 2) ϵ must be at least the standard deviation of the noise density, σ_n (σ_{x_n} and σ_{y_n}) in the spatial domain in terms of (x, y) coordinates for zero mean distribution; and 3) MaxPts must count *less* points than ϵ neighborhood of desired cluster points, $N_\epsilon(q)$, a condition that is directly related to the standard deviation, σ_{cluster} ($\sigma_{x_{\text{cluster}}}$ and $\sigma_{y_{\text{cluster}}}$) of (x, y) coordinates of a dense cluster, and can be chosen based on a priori analysis of the input dataset. These conditions dictate that point noise particle p_n does not belong to the ϵ neighborhood of true clusters N_ϵ, but to Z_ϵ:

$$\begin{cases} p_n \in Z_\epsilon \\ p_n \notin N_\epsilon \end{cases} \quad \text{and} \quad \begin{cases} \text{MinPts} = 1 \\ \text{MaxPts} < \sqrt{(\sigma_{x_{\text{cluster}}} + \sigma_{y_{\text{cluster}}})} \\ \epsilon \geq \sqrt{(\sigma_{x_n} + \sigma_{y_n})} \end{cases}$$

density reachability parameter constraints

A necessary condition for this is that, if a probability distribution of points is defined on the 2D image plane in dataset D as $P(x, y) = \iint_D$, the density distribution of the desired particles, $P(x, y)_{\text{cluster}}$, is higher than the density distribution of the noise, $P_n(x, y)$; otherwise the true clusters will dissolve in the noise:

$$P(x, y)_n < P(x, y)_{\text{cluster}} \tag{19}$$

If the above condition is not met, the clustering will fail for the given condition of dataset D. What this clustering model will do, in essence, is detect the group of desired clusters as points surrounded by *too many other points* (filtered by the max MaxPts conditions) and reject them as noise. The actual noise particles will meet the core noise particle condition of DBSCAN^{-1} as they lack a distinctive concentrated distribution.

A visual representation of this process and the relevance of the DBSCAN^{-1} (in particular in the absence of morphological operations) can be found in Sec. IV.

E. Radial Sorting

Obtaining the cluster center locations in the frame after the clustering routine provides only a static map of the markers, without a spatial orientation with respect to the underlying geometry. To obtain a geometrical representation behind the detected clusters, a radial sorting algorithm is proposed in the processing routine. This algorithm represents the cyan block in Fig. 3. The algorithm is initiated by finding the centroid $\bar{\boldsymbol{c}}_{\text{cp}}$ of the cluster centers (a cloud of points) $\boldsymbol{P}(x, y)$, then obtaining a radially sorted distribution, a so-called convex radial hull, $\boldsymbol{P}_{\theta\text{hull}}$, of n indices, such that the outline of the hull has a continuous connectivity.

First, the algorithm takes as input an arbitrarily indexed cloud of cluster centers, $\boldsymbol{P}(x, y) \in \mathbb{R}^{2\times n}$. The centroid of $\boldsymbol{P}$, $\bar{\boldsymbol{c}}_{\text{cp}}(x, y)$ is calculated to obtain the vector pointing toward the centroid. If the input is a continuous shape, the centroid is sampled at the contours of the area; otherwise for a collection of n points,

$$\bar{\boldsymbol{c}}_{\text{cp}} = \frac{1}{n}\sum_{i=1}^{n} p_i \text{ and} \tag{20}$$

$$\boldsymbol{d}_{\text{cp}} = \boldsymbol{P} - \bar{\boldsymbol{c}}_{\text{cp}} \tag{21}$$

Next, the angle defined by the direction of each vector is calculated and the resulting vector of angles is radially sorted around $\bar{\boldsymbol{c}}_{\text{cp}}$ in the given orientation to obtain the convex radial hull $\boldsymbol{P}_{\theta\text{hull}}$:

$$\boldsymbol{\theta}_{\text{cp}} = \arctan 2(\boldsymbol{d}_{\text{cp}}), \quad \text{where for each point} \tag{22}$$

$$\arctan 2(p)_i = \arctan 2\left(\frac{p_y}{p_x}\right)_i \tag{23}$$

where p_x and p_y indicate the pixel locations in x and y, respectively. Then the sorted index of angles is obtained from sort($\boldsymbol{\theta}_{\text{cp}}$), and the convex radial hull is obtained from sampling by this sorted index:

$$\boldsymbol{P}_{\theta\text{hull}} = \text{sort}\Big(\boldsymbol{P}, \text{sort}(\boldsymbol{\theta}_{\text{cp}})\Big) \tag{24}$$

This is required to draw the outline of points in a continuously connected area. The process of radial hull sorting is shown in Fig. 9.

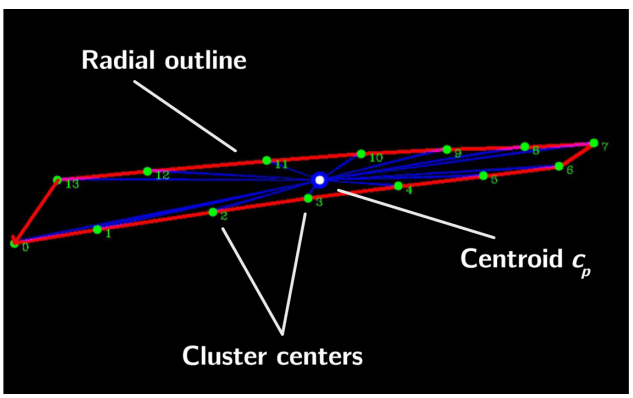


Fig. 9 Radial sorting algorithm process. The cluster centers are green dots, and the convex radial hull is the red outline.

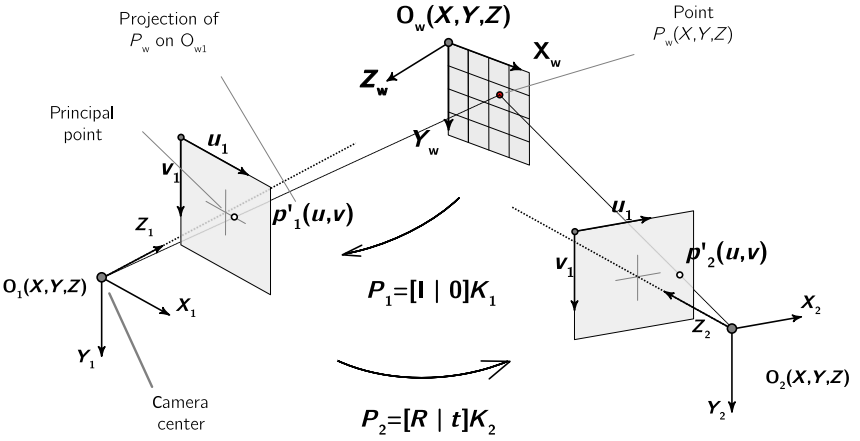


Fig. 10 Schematic of the stereo camera setup and the coordinate systems for 3D reconstruction.

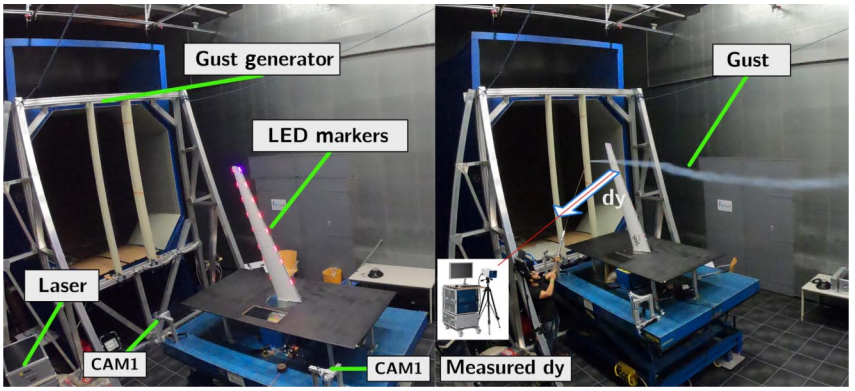


Fig. 11 The Open Jet Facility, showing the gust generator mounted in front of the test section.

Algorithms such as Jarvis march [34] also use a form of radial sorting to wrap a cloud of points in a convex hull. The main difference with the Jarvis march is that the radial sorting algorithm is intended for obtaining a continuous geometry by sorting *all* cluster centers through a continuously connected outline. With a convex hull, some cluster centers may fall inside the region of the convex hull and hence be excluded from the outline. The other difference is that Jarvis march is done at a complexity of $\mathcal{O}(nh)$ (n points and h hull corners), whereas in the proposed approach the sorting can be done in one pass at $\mathcal{O}(n)$ complexity.

F. Reconstruction

The reconstruction is the final step that relates the displacements of corresponding markers in two frames and reconstructs the 3D displacement. The reconstruction process can be inferred from the schematic of the camera setup shown in Fig. 10. Further details regarding the 3D reconstruction can be found in a previous study by Mkhoyan et al. [15,35].

III. Experimental Setup and Data Collection

The experimental data were collected from camera observations of a flexible wing undergoing gust excitations equipped with active LED markers. This experiment was performed within the scope of a larger study on smart sensing methods for control of flexible aircraft.

A. Apparatus

The experiment was conducted in the OJF at the Delft University of Technology [36]. The OJF, as shown in Fig. 11, is a closed-circuit low-speed wind tunnel, driven by a 500 KW electric engine, with an octagonal test section of 285×285 cm^2. The maximum flow velocity available in the wind tunnel is 35 m/s; however, the theoretical performance limit is around 30 m/s.

A gust generator composed of two servo-controlled foam wings was installed in the test section to facilitate various dynamic motion conditions during the test. This particular gust generator allows gust vane deflections of $|\alpha_g| \leq 15°$, or 10°, depending on the actuation frequency (5–7 Hz or 10–15 Hz), and can produce harmonic signal, as well as sweep signals of varying frequencies.

A Polytec Scanning Vibrometers (PSV)-500 laser vibrometer system [37] with a resolution (RMS)[§] of 200 μm/s was used to measure the dynamic response of the wing to the aerodynamic loads introduced by the gust onsets. The PSV system was configured to measure 8 (active) markers, as shown in Fig. 13a, from a total of 16 LED markers placed on the wing. The numbering of the marker IDs in the image tracking algorithm is indicated with curly braces and the laser tracking system in square brackets. Because the laser allowed for measurement of only a single point for each run, each run would be repeated eight times to reconstruct the displacement field of the wing. The system was configured for a sampling rate of 400 Hz.

As shown in Fig. 11 and, schematically, in Fig. 2, a pair of cameras were used to observe the motion of the wing. These cameras are referred to as leading-edge camera (Cam 1) and trailing-edge camera (Cam 2), respectively.

B. Wing Model and Motion Conditions

The wing used in the experiment, referred to as the *Allegra wing*, is a forward-swept tapered wing built of glass-fibre-reinforced plastic. The design of the wing allows for large tip displacements, up to 20% for 10° of angle of attack and 50 m/s flow velocity [38]. The wing was clamped on one side on a sturdy table under a fixed angle of attack of 4°. Detailed information about the wing can be found in Appendix A.

The wing was equipped with 16 LED markers. Each LED marker consisted of three sub-LED units, providing three distinct bright light

[§]Noise-limited resolution is defined here as the signal amplitude root mean square (RMS), measured on a reflective tape, at which the signal-to-noise ratio is 0 dB with 1 Hz spectral resolution.

Table 1 Flow and motion conditions for runs R1, R2 (discrete gusts), and R3 (sweep)

Run ID	Frequency [Hz]	Vane angle [°]	Flow velocity [m/s]	N images	N gusts
R1	5	10	30	469	3
R2	5	5	30	469	3
R3	0.1–10	5	30	574	— —

sources per marker. In the experiment, a 1-cos gust signal and a frequency sweep signal were used.

The data were collected for three experimental conditions or runs denoted R1, R2, and R3, as listed in Table 1. The experimental variables are the flow velocity in the wind tunnel, V_{inf}, the gust vane frequency f_g, and gust vane angle α_g.

For all runs, the images were first recorded in *dark* conditions (night visibility), meaning that the lighting conditions were low for good visibility of the LEDs. Additionally, *bright* images (daylight visibility) were collected to study the effect of HSV filtering in high visibility conditions.

The gust generator parameters were selected such that the disturbance produced a high dynamic response from the wing, to ensure sufficient pixel activity in the image. The gust vane frequency of 5 Hz was close to the wing's natural frequency at the given mass configuration. Runs R1 and R2 each contained three consecutive gust inputs; run R3 did not have a discrete gust, but a sweep signal. The purpose of R2 run was to act as a control against the results of R1, whereas R3 was designed to show marker loss (LEDs on/off) under high dynamic activity.

C. Dataset Collection

1. *Measured Wing Response*

The time history signals in Fig. 12 correspond to the measurements taken at the location of marker ID 1; Figure 12a shows the response of the wing to a single gust input; Fig. 12b shows the response to a sweep signal. The labeling of the marker IDs for the vibrometer measurement system can be found in Fig. 13a. The blue curves in Fig. 12 correspond to the measurements taken by the laser vibrometer sampled at 400 Hz; the red curves are spline models of this response sampled at the capture intervals by the leading-edge camera. The spline model is required to obtain synchronized measurement points between the laser vibrometer data and the image sequences for comparison. The camera images were collected at approximately 40 Hz, with the Nyquist frequency well above the expected resonance frequency of the wing of ≈5 Hz.

2. *Hardware Setup*

An overview of the data acquisition hardware is shown in Fig. 13b. The dataset was recorded with two GigE acA1300-75gc Ethernet Basler cameras with 1300 CMOS 1.3 megapixel (1280 × 1024 pixels) sensor [39]. The cameras were equipped with Computar 12 mm F1.4 2/3″ P IRIS lenses [40] and were positioned in a stereo setup to observe the markers from two viewpoints. The resulting image was cropped to 1088 × 600 pixels and streamed in three-channel RGB format synchronously via real-time precision time protocol (PTP) triggering protocol over the Ethernet. A power over ethernet (PoE) smart switch GS110TP from NETGEAR provided both the power, 3.5 W (per camera unit), as well as the GigE capability to stream the images up to 140 FPS.

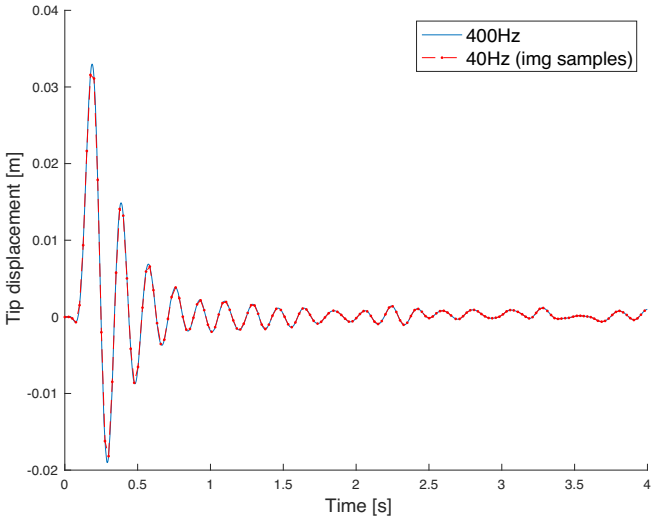
The processing power and image capture during the experiment were delivered by an embedded computing system from NVIDIA, the Jetson TX2, equipped with NVIDIA Pascal architecture with 256 NVIDIA CUDA cores and 1.3 TFLOPS (FP16), dual-core Denver 2 64-bit CPU and quad-core ARM A57 complex [41]. The Jetson TX2 is designed for embedded applications using Artificial Intelligence (AI) and Computer Vision (CV) and operates on Ubuntu 16.04 LTS, allowing flexibility in code deployment. The application developed for this study was programmed in C++ and deployed on the device. For the development the algorithms Basler C++ Pylon API [39] and the OpenCV open-source computer vision library [42] were implemented. To perform image segmentation, image capturing, and compression, GPU hardware acceleration [43] was used with Jetson TX2 dedicated GStreamer pipelines [44].

Code development, algorithm testing, and assessment were done using CPU processing, with a standard Dell Optiplex 7400, a 2.3 GHz Intel Core i5 16G MacBook and the Jetson TX2. The image and tracking data were extracted and plotted using the OpenCV-MATLAB parsing interface tmkhoyan/cvyamlParser [45]. The code, dataset, and tools developed are available under tmkhoyan/adaptive-ClusteringTracker [46].

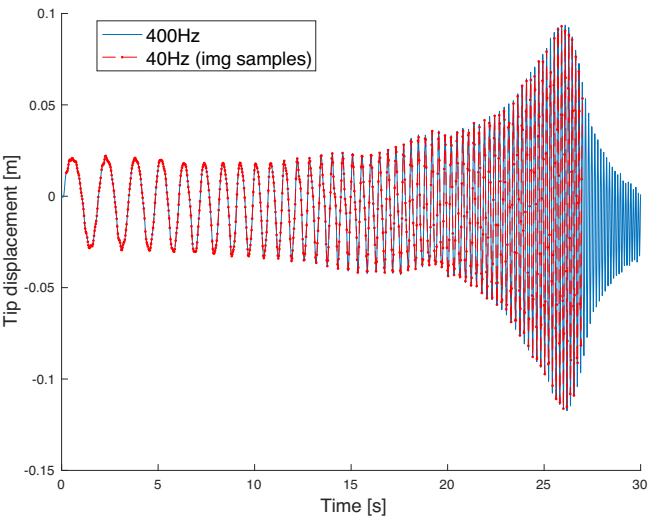
D. Validation Dataset

An automatic labeling tool was developed and implemented to create a reference dataset of the image sequence R1 from Table 1, in order to perform a comparative assessment of the two clustering methods, DBSCAN and disjoint-set data structure.

The tool allows automatic tracking and labeling of the pixelwise (x, y) location of the markers through a sequence of dynamic images, given an initial hand-labeled marker set in the first image. The capability to track a sequence of images classifies it as a tracking routine. However, each consequent frame is visually checked before

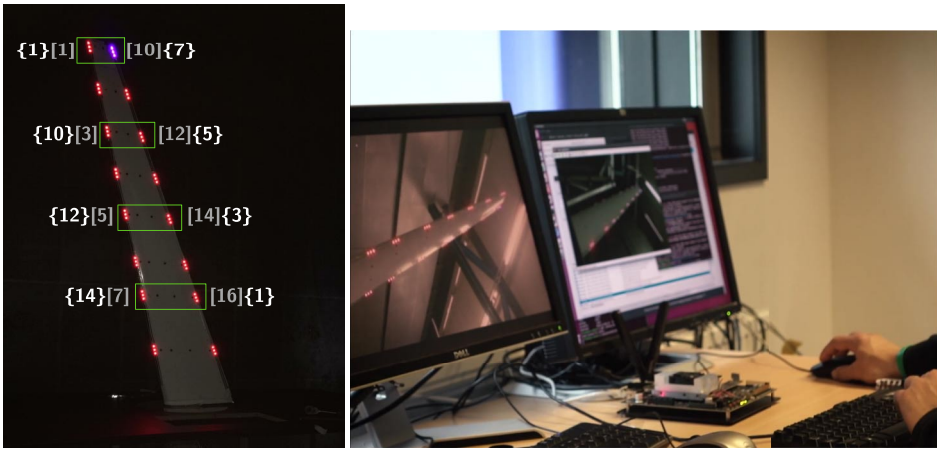


a) Gust signal (1-cos): $\alpha_g = 10°, f_g = 5$ Hz, $V = 30$ m/s



b) Sweep signal: $\alpha_g = 5°, f_g = 0.1 – 10$ Hz, $V = 30$ m/s

Fig. 12 Laser vibrometer measurement (blue line), of the tip displacement of marker ID 1, sampled at capture intervals of Cam 1 at ≈40 Hz (red line); runs with discrete gust (left) and sweep signal (right).



a) The actively measured markers **b) Data acquisition and live tracking hardware setup**

Fig. 13 The experimental setup showing the active markers measured by the Polytec measurement system (in green) and the Jetson TX2 hardware setup.

the labeled data are saved in order to ensure the validity of the reference dataset. The processing strategy of the tool can be summarized in the following way:

1) Manually select initial marker locations of the first image in the sequence. The marker locations are defined as a (x, y) pixel location of the center LED of each three-LED marker cluster.

2) A submatrix is defined as a bounding square box enclosing the three-LED cluster, at the (x, y) location of the center LED, with a width of 40 pixels.

3) The submatrices corresponding to the number of markers (14 in total for the R1 image sequence) serve as an input to the automatic detection of the markers in the next image frame.

4) A detector is implemented to process each submatrix, defined at the location of the submatrix from the previous image with an uncertainty factor of 1.2 in width and height (i.e., the bounding square is factor 1.2 larger than the initial submatrix).

5) Each automatic detection is visually approved before moving to the next frame and saving the data. The uncertainty margin (1.2 factor in width and height) is implemented such that the new, shifted marker location can be found with respect to the previous image, and enough margin is kept to account for the motion. This process is depicted in Fig. 14.

The tool enables the implementation of a custom detector for detection of the circular LED markers. In the current study, a contour filter, often referred to as a blob detector, was used based on the Topological Structural Analysis algorithm of binary images and shapes [27]. Before the detection, the submatrix thresholding is applied using Otsu's adaptive thresholding method, such that a binary mask of the marker outline is obtained. This tool was developed in C++ programming language using the OpenCV open-source computer vision library [42], and made available under the BSD-3 license [47].

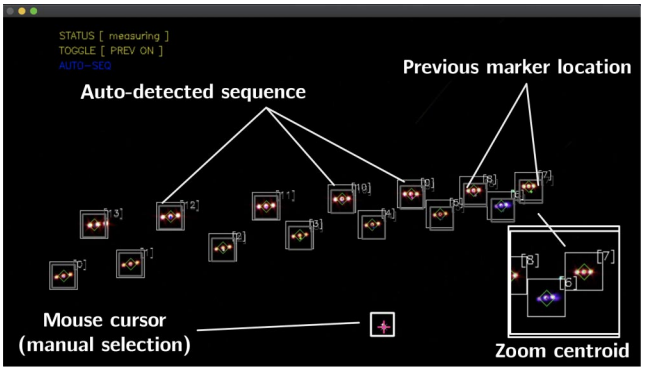


Fig. 14 The labeling process with the automatic labeling tool.

E. Clustering Performance Test

To compare DBSCAN to the disjoint-set data structure, a performance test was designed. In this test a grid, $I(x, y)_{\text{grid}}$ of 10,000 × 10,000 pixels was used, and clusters of particles were generated randomly to perform the clustering. For each run, the grid was initialized with a varying number of cluster centers, m_i (e.g., 10, 50, 100), with a uniform distribution. The grid size is used as the minimum and maximum bounds of this distribution, with a 0.9 shrink factor to keep 10% free at the borders. The cluster center distribution is defined as follows:

$$P_{\text{centre}} \in I(x, y)_{\text{grid}}, \text{ and } \begin{cases} x_{\min} = 0.1 \cdot w_{\text{img}}, & x_{\max} = 0.9 \cdot w_{\text{img}} \\ y_{\min} = 0.1 \cdot h_{\text{img}}, & y_{\max} = 0.9 \cdot h_{\text{img}} \end{cases} \tag{25}$$

Around these m_i cluster centers, a fixed number of $n_i = 50$ scatter points was sampled with a normal distribution with the following properties in both x and y locations:

$$P_{\text{cluster}} \in I(x, y)_{\text{grid}}, \text{ and } \begin{cases} \mu_{\text{cluster}} = \mu_{\text{cluster}_x} = \mu_{\text{cluster}_y} = 0 \\ \sigma_{\text{cluster}} = \sigma_{\text{cluster}_x} = \sigma_{\text{cluster}_y} = h_{\text{img}}/100 \end{cases} \tag{26}$$

Here, the sampled normal distribution, P_{cluster}, is the offset from a cluster center (x, y) pixel coordinate; μ_{cluster} and σ_{cluster} are the mean and standard deviation of the distribution. The resulting scatter model is a cloud of points, with the majority falling inside a radius of σ_{cluster} (defined as a factor of image width, h_{img}) from the cluster center. Figure 15 shows a randomly sampled dataset with $m_i = 10$ number of cluster centers and cluster size of $n_i = 50$. To assess the clustering methods on their ability to cope with noise, for each run, uniformly distributed noise was generated on top of the existing points. These scattered noise points, n_{inoise}, were proportional to the number of cluster centers with a factor 5 (i.e., $n_{\text{inoise}} = m_i \times 5$).

The performance test was used to generate the performance dataset; the code was developed in C++ programming language using the OpenCV open-source computer vision library [42], and made available under the MIT license [48].

F. Noise Model

A common noise model was used for evaluating the real-life performance of the clustering methods. The input images were injected with an image-independent Gaussian noise, and the robustness of the color filtering, thresholding, and clustering pipeline was investigated against possible sensor noise, transmission, and hardware-related issues and poor illumination. Subsequently, the tracking quality of the pipeline was assessed on image sequences from

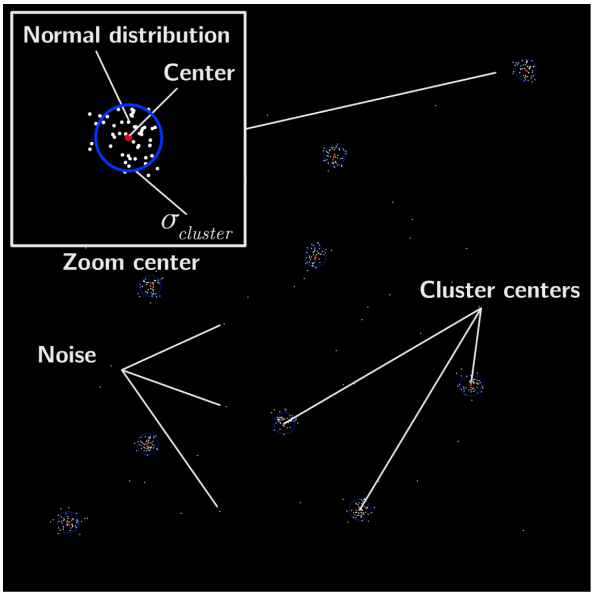


Fig. 15 Randomly sampled scatter data for 10 cluster centers and a cluster size of 50.

R1 and R3, whereas R2 was used as a reference. The probability density function of the Gaussian noise model is as follows:

$$I(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{(z-\mu)^2/2\sigma^2} \tag{27}$$

In this model, z represents the grayscale value. The parameters used for the noise model are a mean of $\mu = 0$ and a standard deviation of $\sigma = 0.5$. The gray values produced from the probability distribution are scaled to RGB range 0–255 and injected in the three channels of the input image $I(x, y)$, producing a new additive noise input image $J(x, y)$. The random seed is initialized with the CPU clock (time t) for each image input, $I(x, y)_k$, resulting in a dynamic noisy image input sequence, $I(x, y)_{\text{noise}_k}$, at each kth frame:

$$I(x, y)_{\text{noise}_k} = I(x, y)_k + N(x, y, t) \tag{28}$$

Here, $N(x, y, t)$ is the random seed initialized noise mask.

IV. Results and Discussion

The experimental data were processed with two clustering pipelines implementing DBSCAN and disjoint-set data structure clustering methods. Here, a distinction must be made between the randomly generated *performance dataset* generated with the performance test discussed in Sec. III.E and the *experimental dataset*, as provided in Table 1.

The performance test was done with the purpose of extracting the isolated clusters of points from scattered data. Here, the novel implementation of DBSCAN was used, with the additional MaxPts parameter set to 100, MinPts to 20, and ϵ to 180 pix. The ϵ parameter was chosen to have a value approximately twice as large as the standard deviation of the cluster population, σ_{cluster}, in order to capture the majority of the randomly generated cluster points scattered around the cluster centers. For the disjoint-set data structure, the distance parameter γ was chosen to be equal to ϵ.

Furthermore, the tracking result with the full clustering pipeline on the runs R1 and R3 were performed on a sequence of ≈469 images from Cam 1 (leading edge). R2 was used as the control for R1 and showed a similar result. R3 was mainly used to assess the ability of the tracking pipeline to deal with marker loss. In the nominal runs, $\epsilon = 20$ pix and the reachability parameter MinPts $= 2$ were used. This set of parameters provided the best cluster detection considering preceding segmentation filters. Here, as in the previous case, the distance parameter γ for the disjoint-set data structure was chosen equal to ϵ of the DBSCAN.

Speeds of 250+ FPS were measured for the DBSCAN implementation, on an image sequence of a single camera with a resolution of 1088×600 pixels using a standard Dell Optiplex 7400, a 2.3 GHz Intel Core i5 16G MacBook, and the Jetson TX2. The outcomes of both methods were compared with the reference data collected by the automatic labeling tool that was developed explicitly for this purpose, as addressed in Sec. III.D.

A. Performance Test of DBSCAN and Disjoint-Set Data Structure Clustering Methods

The performance test was executed with cluster center sizes $m_i = 5, 10, 50, 100, 200, 500$ and cluster population sizes of $n_i = 50, 50, 50, 50, 50, 50$. For cluster center sizes $m_i < 100$, the uniform distribution of the cluster centers was balanced to ensure a minimum distance from each cluster center. This was done in order to prevent cluster populations from merging. For larger cluster center sizes ($\geq$200) merging was allowed.

The results of the clustering are shown, from left to right, for cluster center sizes $m_i = 10, 100, 200$ in Fig. 16. The purple and yellow radii and their respective centers represent the detected clusters and correspond to the ϵ and γ of the DBSCAN and disjoint-set data structure, respectively. For all runs, the advantage of DBSCAN with regard to noise is evident. Even in the presence of relatively low noise ($n_{i_{\text{noise}}} = 50$ for $m_i = 10$), the disjoint-set data structure fails and, aside from real clusters, also classifies these noise particles as clusters. DBSCAN, on the other hand, can make this distinction and extract the correct number of isolated clusters. As the population density increases, the initial scatter distribution is not balanced, and certain clusters merge; therefore, the number of detected clusters does not have to correspond to the number of initial clusters. For the remaining two runs, DBSCAN is consistent in performing the task and is able to separate and correctly identify the isolated clusters.

The advantage of the novel DBSCAN formulation and the MaxPts was also evaluated for this particular task. With the addition of the

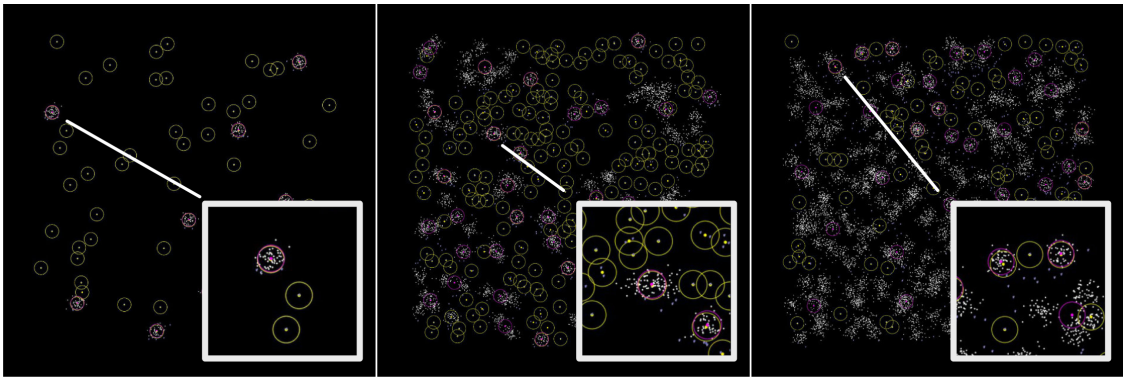


Fig. 16 Results of the performance for (left to right) $m_i = 10, 100, 200$ and $n_i = 50, 50, 50$.

MaxPts, DBSCAN is able to ignore widely scattered clusters and focuses more on isolated islands of clusters.

B. Validation of DBSCAN and Disjoint-Set Data Structure Clustering Pipeline

Two clustering methods were compared and validated against the validation dataset created with the automatic labeling tool. The clustering methods were implemented in the full tracking pipeline, and the image sequence from the experimental dataset R1 was processed. The images were sampled at the sampling intervals (red line), as shown in Fig. 12. Here the laser vibrometer measurement (in blue), showing the motion of the wing, is later compared with the output of the validated tracking pipeline. Sequence R2 showed a similar response to R1 and is therefore not included in the following validation plots.

1. *Spatial Marker Scatter Validation*

Figure 17 shows an overview of the clustering results. Here, the marker location (defined as the centroid of the three-LED contour) in the first image is indicated with yellow, purple, and blue circles corresponding to DBSCAN, disjoint-set data structure, and validation data, respectively. The scattered points reflect the marker positions detected in the complete sequence of R1, where the color is kept consistent for the data. The motion of the wing, more specifically the tip deflection, reflects in the spread of the scatter points observed. The spread is the highest for the markers closer to the tip, as expected.

In the two boxes in Fig. 18, a zoom is shown for maker 9 with the lowest error, and markers 2 and 7 with the highest errors with respect to the validation data. In these figures, the marker location of the initial image at rest is indicated by a purple cross and a yellow plus sign for DBSCAN and disjoint-set data structure, respectively, to distinguish the results of the two methods. The mean is indicated by the diamond and the triangle shapes. Figure 18a shows a relatively low error and the spread is closely packed. In Figs. 18b and 18c, both methods show a larger spread in (x, y) with respect to the validation data. However, disjoint-set data structure has a higher spread, mainly in the x direction.

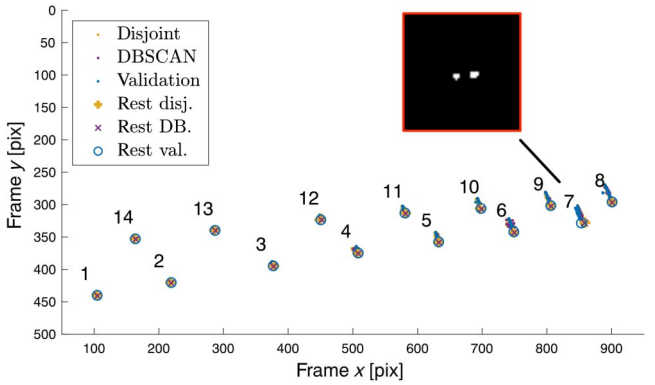


Fig. 17 Spread of tracked markers for run R1 with DBSCAN and disjoint-set data structure.

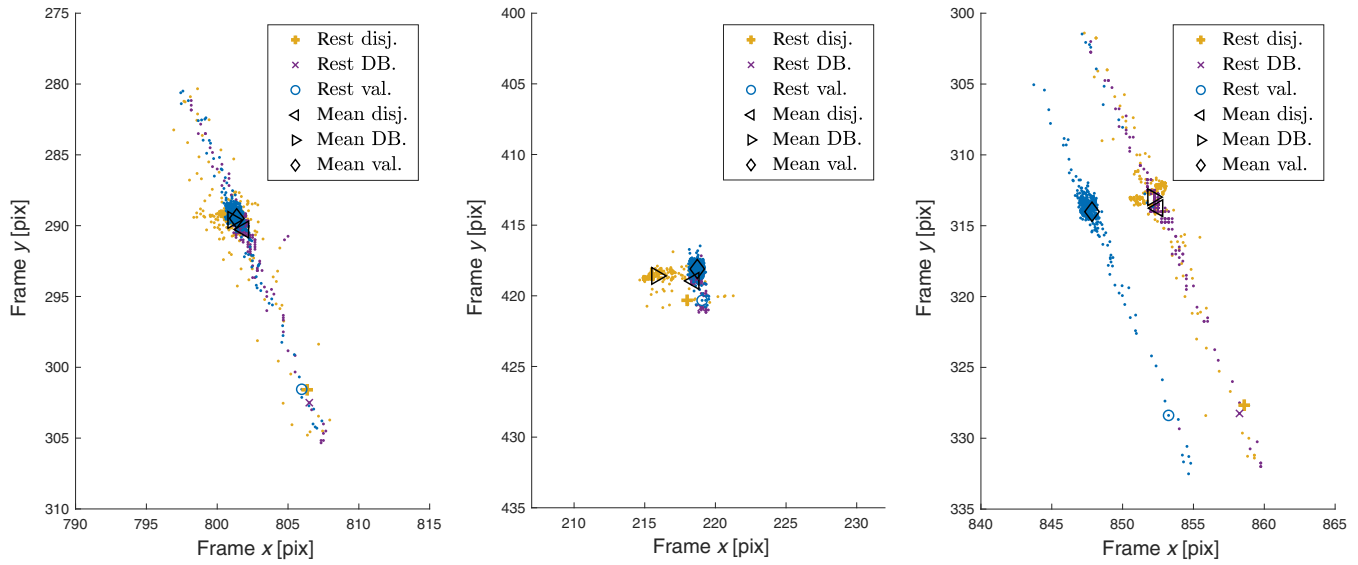
To quantify the error, a squared distance error metric is used close to the formulation of the root mean square error (RMSE). The error is defined as the squared average of the Euclidean distance throughout the sequence i to n:

$$\mathrm{RMSE} = \sqrt{\sum_{i=1}^{n} \mathrm{dist}(p, \hat{p})_{\mathrm{euclid}_i}} \tag{29}$$

where the Euclidean norm of the reference point, $p(x, y)$, and its estimate, $\hat{p}(x, y)$, throughout the sequence i to n yields

$$\mathrm{dist}(p, \hat{p})_{\mathrm{euclid}_i} = \sqrt{\sum_{i=1}^{n} (p(x, y)_i - \hat{p}(x, y)_i)^2} \tag{30}$$

From the boxplot in Fig. 19, a better insight can be gained in the average error of the Euclidean distance norm in x and y [Eq. (29)]. Here, the color codes are, again, consistent with the clustering methods; furthermore, the diamonds indicate the mean of the data and the red crosses the outliers. The outliers are defined as the points that are factor 1.5 larger than the bounds of the interquartile range (i.e., data between 25th and 75th percentiles). It is observed that the average error through the complete R1 sequence lies below 1 pixel for the majority of the markers with DBSCAN. Disjoint-set data structure shows a relatively higher spread and a larger mean error. This is observed in particular for markers 1, 2, and 7. For the latter marker, DBSCAN also shows significantly larger errors of up to 4.5 pixels; however, in markers 1 and 2, disjoint-set data structure has a factor 2 and 3 larger error, respectively. These observations are consistent with the results shown in Fig. 18. The large errors can be attributed to the fact that the contours of the three-LED marker cluster merge together due to the motion of the wing and a slight change in the LED-reflection results in two distinct contour shapes. Therefore, the centroid of these three-LED markers falls approximately 3–4 pixels away from the true centroid. This is visible in the close-up box in Fig. 17.



a) Zoom of marker ID 9 (low error) **b) Zoom of marker 2 (high error)** **c) Zoom of marker ID 7 (high error)**

Fig. 18 Zoom spread markers IDs 2 and 7 of run R1 with DBSCAN and disjoint-set data structure.

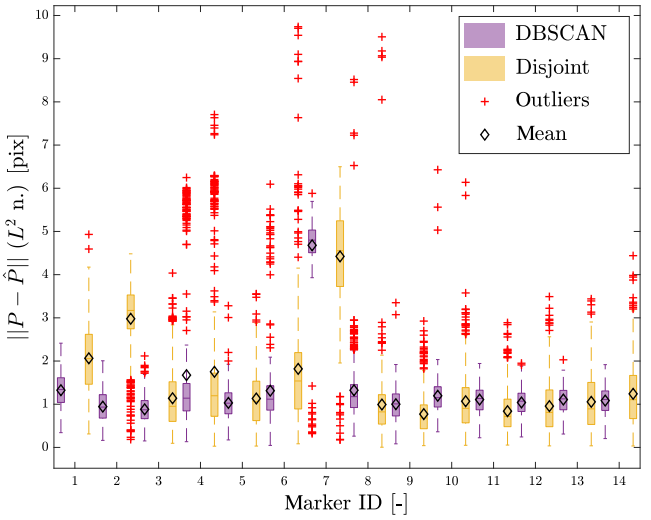


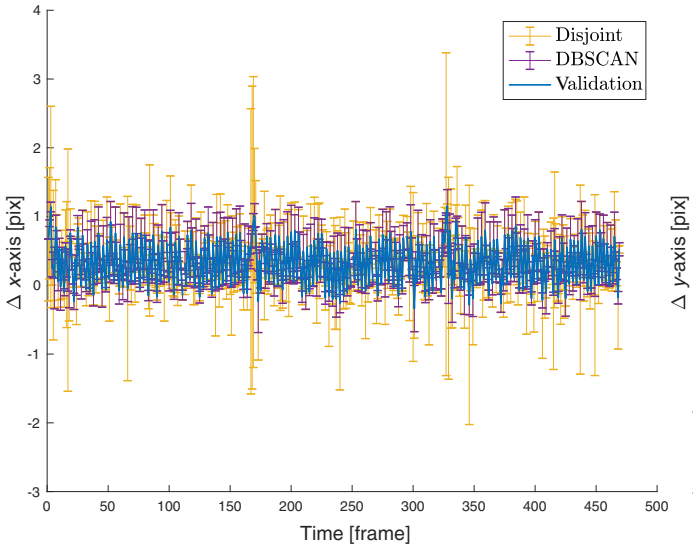
Fig. 19 Euclidean norm tracking error for run R1 with DBSCAN and disjoint-set data structure.

2. Sequence Tracking Validation

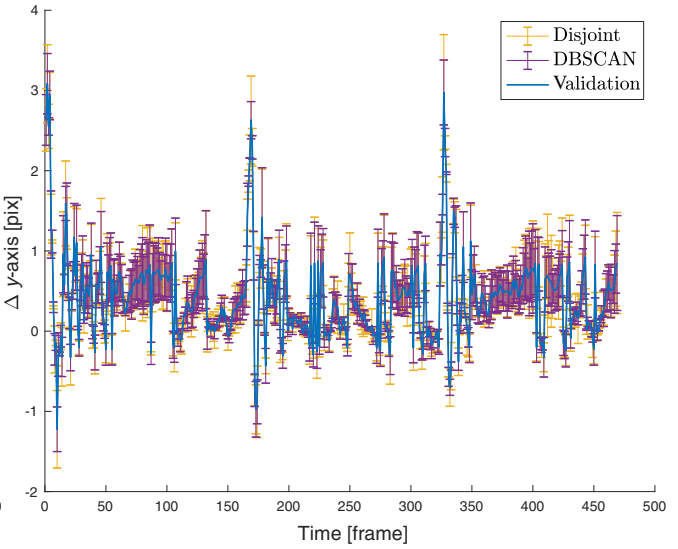
By observing the tracking of the markers 2 and 7 across the R1 sequence, one can confirm the larger errors of the disjoint-set data structure. Figure 20 shows the time traces of the displacement of the markers 2 and 7 in *x* and *y* directions, respectively, with respect to the steady-state position (initial image). In R1, exactly three gusts were introduced, which can be observed by the three peaks followed by decaying sinusoidal responses. Alongside the displacement, error bars are shown, defined according to Eq. (29). Again, the color codes are kept consistent, and the disjoint-set data structure shows a larger error band (up to 3 pixels) compared with DBSCAN and the reference data. In particular, the *x* direction exhibits a higher sensitivity to errors as can be observed in the presented scatter plots. DBSCAN shows better agreement with the validation data.

C. Experimental Data Analysis with DBSCAN and Disjoint-Set Data Structure Clustering

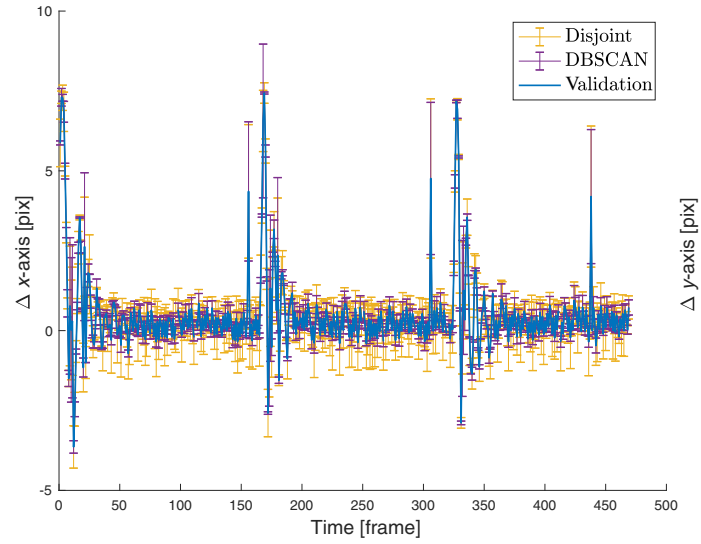
The experimental data collected from the image sequences were processed with both tracking pipelines and compared with the laser vibrometer measured wing response, shown in Fig. 12a. The tracking pipeline, depicted in Fig. 25, was found to be able to trace the motion



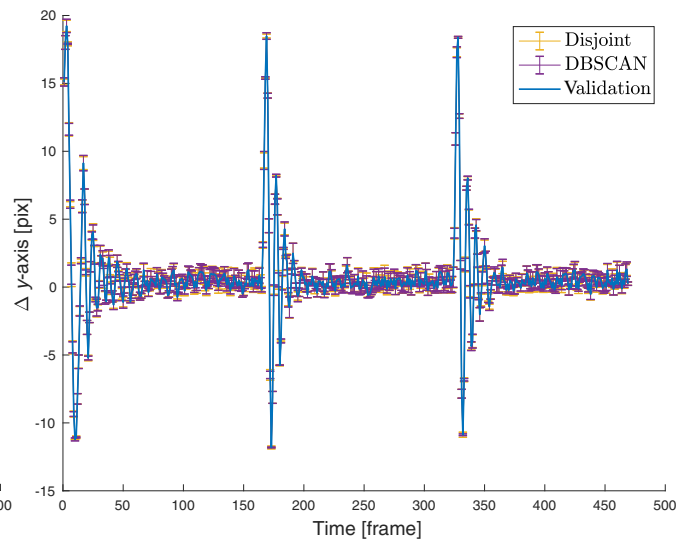
a) Displacement in *x* of R1 sequence marker ID 2



b) Displacement in *y* of R1 sequence marker ID 2

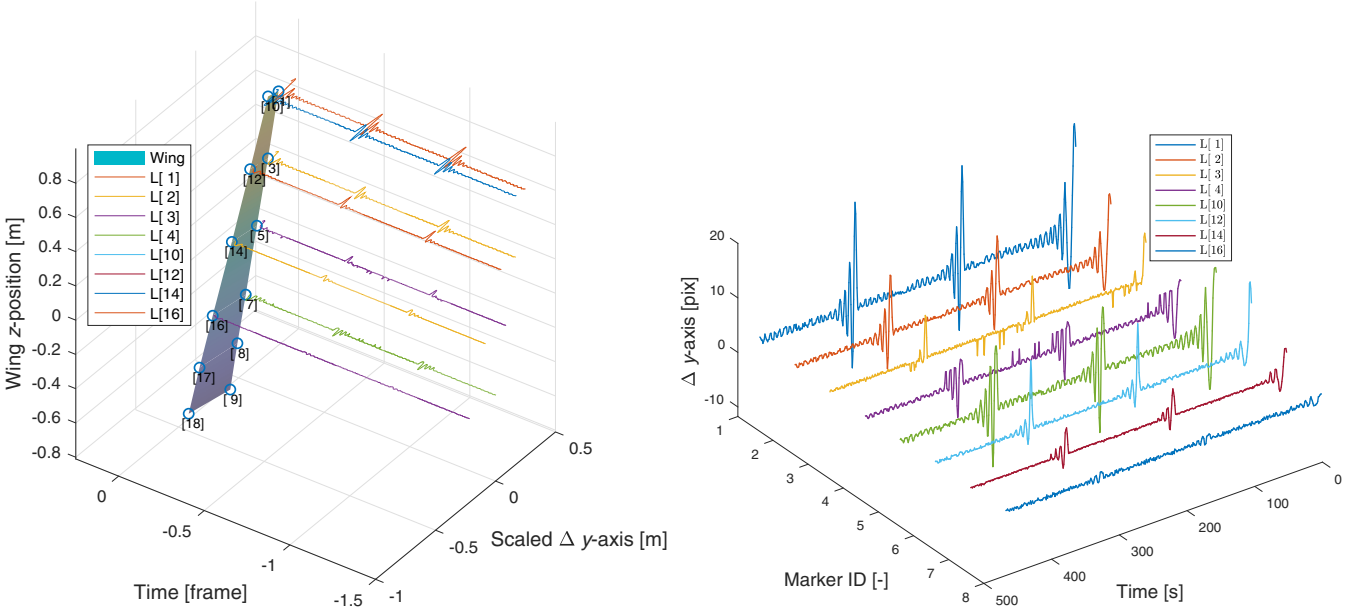


c) Displacement in *x* of R1 sequence marker ID 7



d) Displacement in *y* of R1 sequence marker ID 7

Fig. 20 Validation and comparison of DBSCAN and disjoint-set data structure results for run R1, marker IDs 7 and 2.



a)Wing 3D view and scaled image *y*-displacement **b) Image *y*-displacement relative to the baseline**

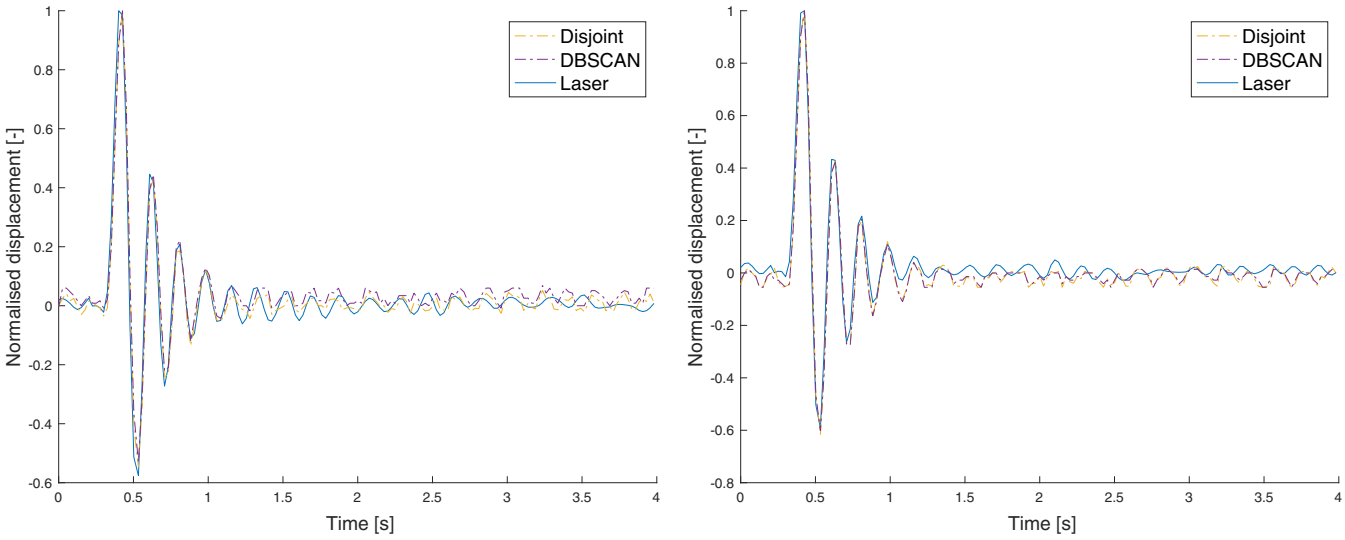
Fig. 21 Time series of marker *y* displacement with DBSCAN across 469 image sequences of run R1.

of the wing by correctly clustering the markers. This was confirmed by the DBSCAN tracking result in terms of pixel (x, y) locations in the images, shown in Fig. 21. The plots show the time traces for the detected markers and three occurrences of decaying sinusoidal responses can be observed from the output. The image *y*-displacement in pixels is arbitrarily scaled to show comparative response to wing geometry. During the image sequence, exactly three gusts were introduced to the wing that produced a measurement, as shown in Fig. 12 for a single gust.

1. *Time-Domain Analysis*

To further quantify the measured wing response and the tracked wing motion from the image sequence, comparisons were made in terms of displacements and the frequency content. Figure 21 shows the comparison of the sampled wing displacement response extracted from the laser vibrometer (blue), the disjoint-set data structure tracking pipeline (yellow), and the DBSCAN tracking pipeline (purple). In this figure, markers with high motion amplitude and a relatively high (marker ID 7), as well as low (marker ID 8), validation errors are compared. The response, shown here for both tracking pipelines, is the displacement in *y* direction, where the highest amplitudes were observed.

The laser vibrometer measured response (in meters), as well as the visually tracked motion (in pixels *y* direction), were normalized to allow effective comparison. The tracked responses in Fig. 22 show good agreement with the laser measurements. Both tracking methods are capable of capturing the inherent damping of the wing with the correct amplitude decay and match the phase of the motion. The tracking results for marker ID 8 are slightly better compared with marker ID 7, in agreement with the higher validation error measured for ID 7, as shown in Fig. 19. However, as the error was mainly observed in *x* direction, differences in *y* are not significant. DBSCAN shows better overall tracking performance. Furthermore, by observing the low-amplitude oscillations after the initial gust onset, it can be seen that image tracking has a limit in terms of accuracy and resolution. In Fig. 22a, it is observed that the tracking reduces in accuracy after the fourth peak (at 1 s) at roughly 10% of the maximum normalized amplitude, and the motion is overestimated by the tracking (from 2 s) by roughly 3% in the worst case. Considering



a) Comparison normalised response marker ID 8 **b) Comparison normalised response marker ID 7**

Fig. 22 Comparison of normalized response of the tip deflection for run R1 (laser vs image tracking).

that the maximum tip deflection is ≈33 mm (Fig. 12a), the maximum motion tracking resolution is in the order of ≈0.1 mm. An obvious way to increase this resolution would be to use higher-resolution cameras (>1.3MP). However, this could directly increase the computational cost of the pipeline and reduce the maximum processing frame rate, with a subsequent penalty for the bandwidth of the image tracking.

2. Frequency-Domain Analysis

Alternatively, a frequency domain analysis was performed, where the measurements were compared in terms of the power spectral densities (PSDs) of the measured output. The main objective of this analysis was to understand whether the image tracking methods could correctly identify the frequency content of the measured signal compared with the reference measurement provided by the laser vibrometer. In this context, the power spectrum of the same response signal measured by three different methods provided sufficient grounds for the comparative assessment. This eliminated the need for a more elaborate frequency response analysis whereby the cross power spectrum of the output to input signals is computed as well in order to extract the system's frequency response function.

The auto-PSD of the output signal, S_{yy}, was calculated according to the following definition:

$$S_{yy}(\omega) = \int_{-\infty}^{\infty} R_{yy}(\tau) e^{-j\omega\tau}\, \mathrm{d}\tau \tag{31}$$

where the integral in the expression is the Fourier transform of the autocorrelation function R_{yy} of the output signal (marker displacement). Figure 23 shows the frequency content of the image sequence corresponding to the responses of marker IDs 8 and 7. As can be observed from Fig. 23a, the image tracking methods are able to estimate the first resonant frequency of the wing, as for both markers, the peaks of the spectral densities align at 5.316 Hz. Furthermore, it is observed that, despite higher errors in the tracked response of ID 7, the resonance region is captured well in both tracking methods.

Here, the disjoint-set data structure method estimates a slightly higher value for the power spectrum in the resonance region, which may seem to match better the distribution obtained from the laser measurement. This can be explained by observing the response in Figs. 22a and 22b, showing that the oscillations tracked by the disjoint-set data structure have the tendency to estimate a higher power distribution around the resonance region compared with DBSCAN (i.e., higher sensitivity gain toward motion). For some markers, in particular, the ones that exhibit lower motion activity (markers closer to the root), this can lead to overestimation of the response and (combined with a higher error) a shifted resonance peak. This is visible in Fig. 24 for marker ID 3. Here, the disjoint-set data structure overestimates the oscillations of the y displacement (Fig. 24a) and the resulting resonance peak is shifted from 5.316 to 4.810 Hz (Fig. 24b). Although the RMSE of marker ID 3 is lower compared with ID 7 (Fig. 19), the high RMSE of marker ID 7 is largely contributed by the x displacement; hence for marker ID 3 a higher error in the y displacement is probable. DBSCAN also has a slightly higher error for marker ID 3 but estimates the peak more accurately at 5.570 Hz.

Overall, the results are shown in Figs. 22 and 23 suggest that the motion of an oscillating wing can be captured and analyzed with relatively low-resolution cameras (1.3 megapixels). However, it is preferred to use markers exhibiting high motion amplitude (closer to the tip).

3. Clustering Image Sequence

The clustering results of sequence R3, containing the marker loss due to LED failure, are shown in the lower row of Fig. 25. Here, the dotted outline shows the initial contour at baseline deflected shape before the gust hits the wing. The tracking pipeline schematic is shown below the figure. Despite the marker loss, DBSCAN is able to correctly deduce the number and the location of the markers, without supervision in terms of the expected number of clusters.

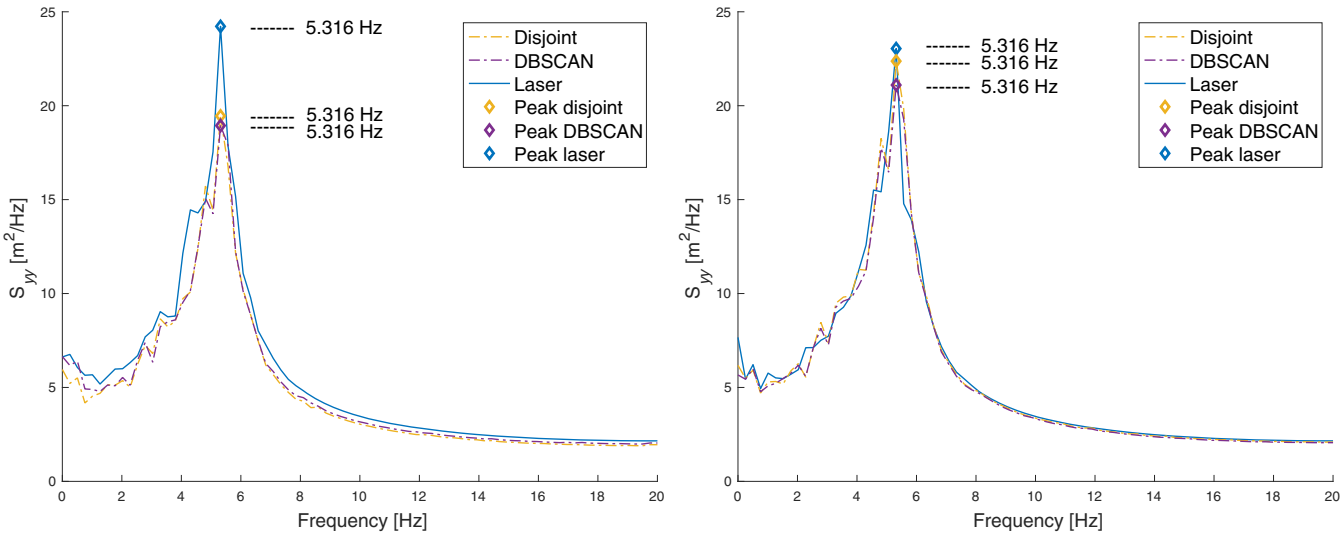
D. Performance of DBSCAN^{-1} and the Limitation of DBSCAN Clustering

1. Assessment of DBSCAN Parameters

Figure 26 shows the sensitivity of the tracking results to the MaxPts parameter. When the parameter is set to 3, dictating that the direct density reachability of the core points needs to contain a neighborhood of at least three core points, the markers 1 and 10 fail to meet these criteria and are no longer considered to be core points. The shapes (dataset D) in the extracted binary mask on which the clustering operation is done are influenced by the motion of the wing and the result of morphological filters (erode, dilate) performed after HSV filtering. As a result, at a given time instance, the three-LED subunits can be clotted together in one or two dots instead of three, never meeting the core point condition. In Fig. 26, it is clearly illustrated that the cluster is found again once the units become more distinct; this is the case when the MaxPts parameter is chosen to be 2, as shown in Fig. 25.

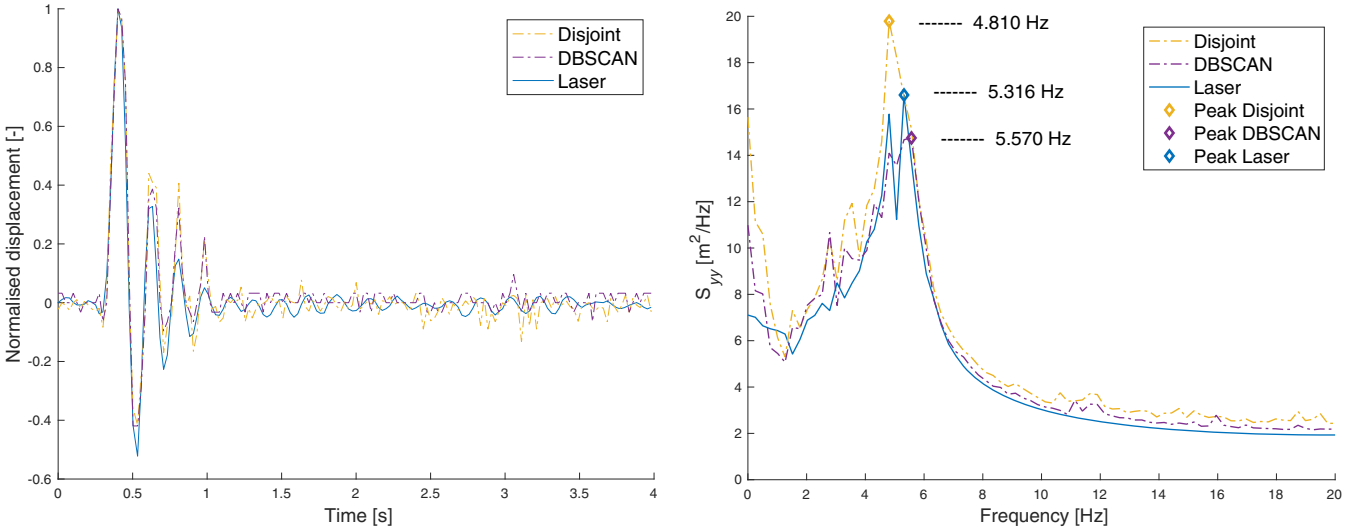
2. Evaluation of Robustness Against Noise

The runs R1 and R3 were injected with Gaussian noise (mean of $\mu = 0$ and a standard deviation of $\sigma = 0.5$) and the performance of



a) Auto-PSD of the response signal of marker ID 7 **b) Auto-PSD of the response signal of marker ID 8**

Fig. 23 Comparison of the frequency spectrum of the tip deflection response for run R1 (laser vs image tracking).



a) Comparison normalised response marker ID 3 **b) Auto-PSD of the response signal marker ID 3**

Fig. 24 Comparison of the response and the frequency spectrum of the marker ID 3, run R1 (laser vs image tracking).

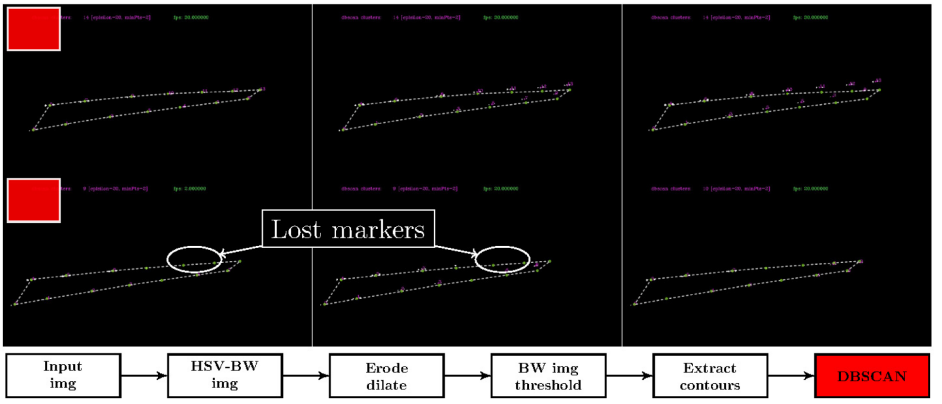


Fig. 25 Tracking sequence on input images from run R1 (upper row) and R3 (lower row).

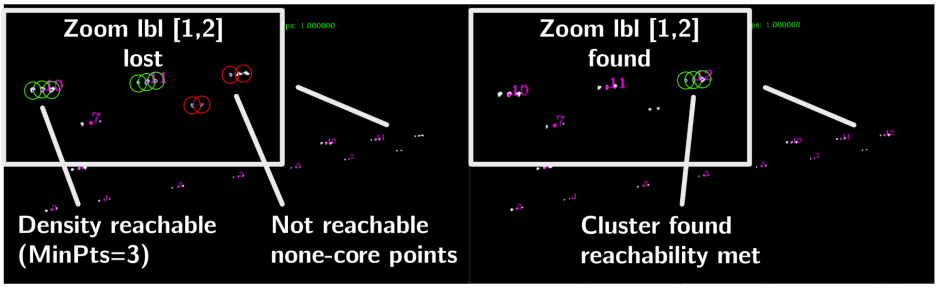


Fig. 26 Sensitivity of DBSCAN parameters. Snapshot of two frames from sequence R1; the frames are ≈0.025 s apart. The parameters are $\epsilon = 20$, MinPts $= 3$.

the tracking was evaluated. In Fig. 27, a sequence is shown for tracking of frames 0, 50, and 100. The schematic of the corresponding tracking pipeline is provided at the bottom of the figure. The color codes correspond to the operation steps performed in the pipeline throughout the sequence. From top to bottom, the rows represent input with noise image (gray), HSV filtering (green), threshold image (blue), and clustering result (red). The dotted outline shows the initial contour at baseline deflected shape before the gust hits the wing.

As can be seen, the HSV filter combined with the morphological operations (erode and dilate) is able to cope well with the Gaussian noise. The morphological operations together with the HSV filter are in fact acting as a complex denoising filter that produces a clean output, which is in turn passed through as an input to DBSCAN. DBSCAN is then able to produce a robust result on the thresholded binary image, despite the high level of noise injected into the input image. For run R3, similar results were obtained, as presented in Appendix B (Fig. B1).

3. *Sensitivity of HSV Filtering and Morphological Operations*

The benefit of the complex denoising filter (HSV morphological) with regard to noise and better clustering of input data was evident. There is, however, a condition where a combination of these filters can have an adverse effect. This is, in particular, the case in the presence of varying lighting conditions, such as the case for images

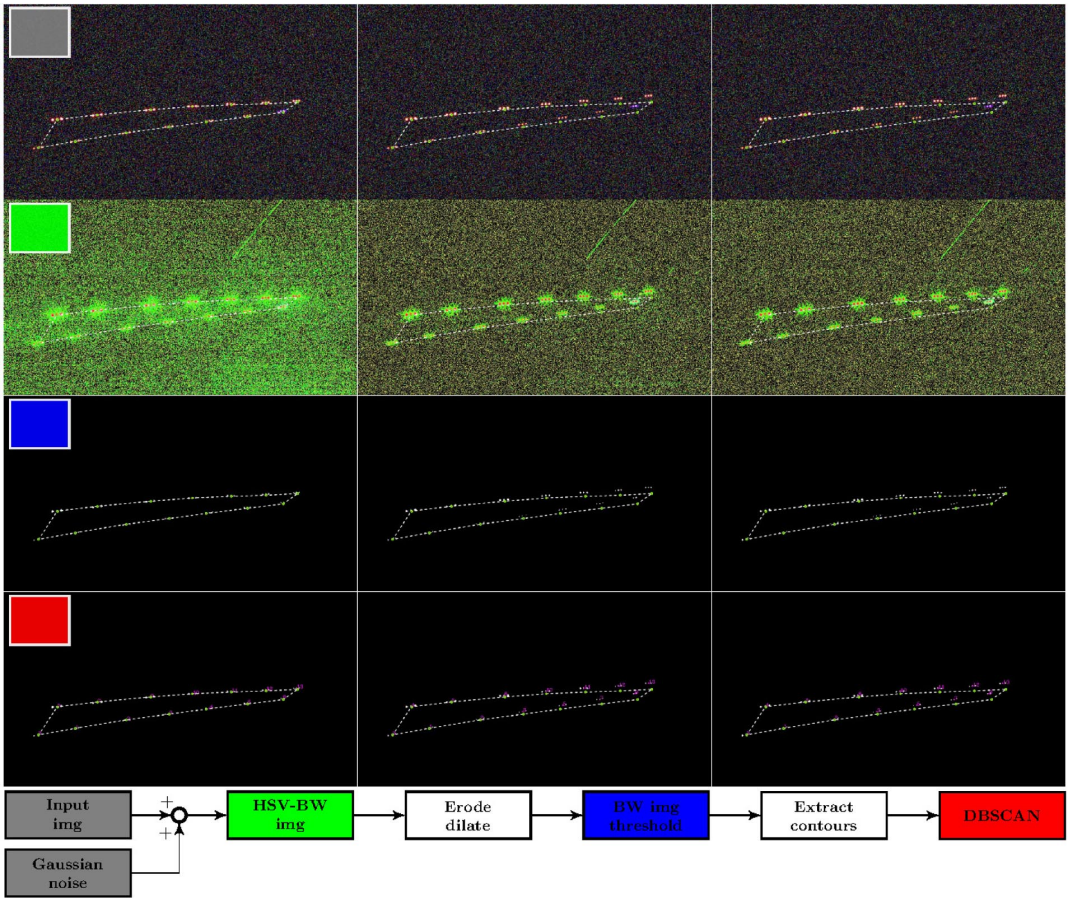


Fig. 27 Tracking sequence on input images (0, 50, 100) from run R1 with injected Gaussian noise ($\mu = 0$ and $\sigma = 0.5$).

with a light source, i.e., bright images, as per the definition in Sec. III.B.

In Fig. 28, the effect of HSV filtering strategies is shown for images with a light source recorded by the trailing edge camera. The columns of images, from left to right, correspond to 1) HSV filtering with morphological operations and no additional noise in the input (default case); 2) same, but without morphological operations; and 3) same as condition 1, but with added noise in the input. The pipeline

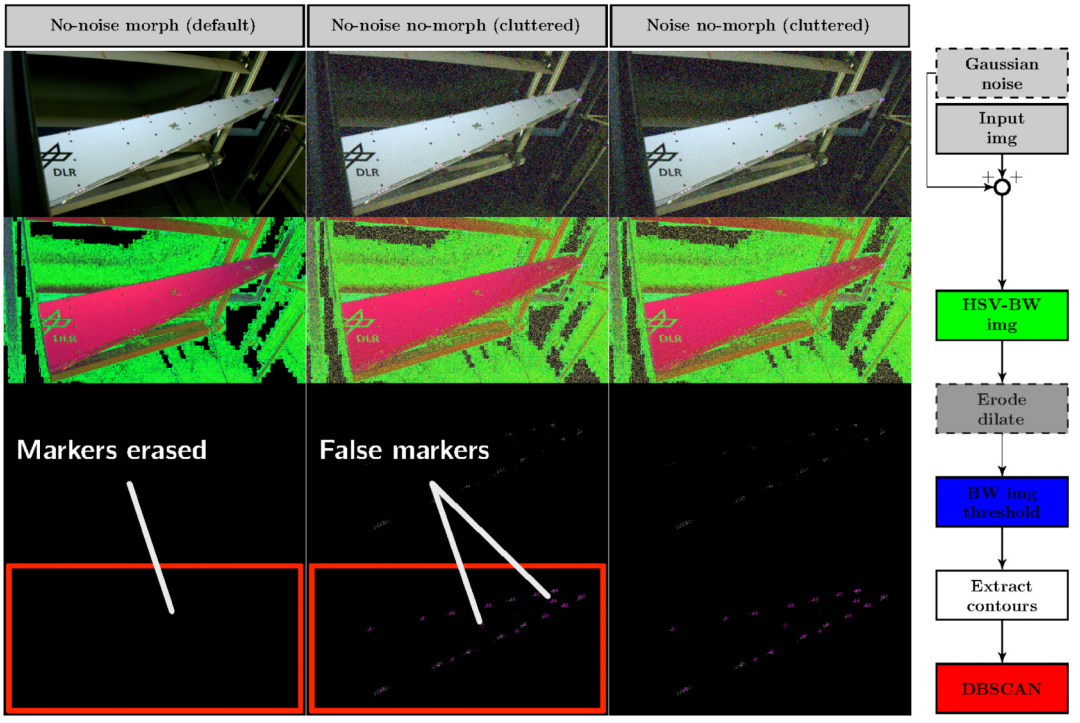


Fig. 28 Sensitivity of HSV filtering and morphological operations to varying lighting conditions.

shown on the right of Fig. 28 corresponds to the processing steps along the rows. As was observed in case 1, the default complex filter had a highly *adverse effect* and the marker contours required for clustering were largely *erased*. This happens as the HSV filter produces high-density scattered noise particles in the output image (the color content in the lighter image is higher), and the subsequent morphological erode filter, due to the local min-max operation, ends up removing the relevant data from the output image alongside the noise. This becomes clear when compared with case 2, where no morphological operations were applied: the contours of the markers are retained, although with much higher noise content compared with the nominal R1 dataset (dark images). This results in false detection of markers (center area of the wing) in the clustering step. When additional noise is added (without morphological operations) in case 3 the result is similar, but noisier, as expected. This analysis indicates that morphological operations are not always possible or beneficial, and highlights the relevance and need for a novel clustering approach with DBSCAN^{-1}.

4. DBSCAN^{-1} *in the Presence of Noise Without Morphological Operations*

In this analysis, the robustness of DBSCAN was investigated without additional denoising filters. The same experiment was run to see how well DBSCAN would fare when exposed to more noise and less filtering steps, and in particular when morphological operations were removed. These operations proved to be highly capable of filtering out the remnants of Gaussian noise after HSV operation, marked as the blue filtering block in Fig. 27. It was therefore interesting to examine what the effect of removing the morphological filter block would be.

The result with the nominal DBSCAN using the same parameters ($\epsilon = 20$ pix, MinPts $= 2$) is shown in Fig. 29. The schematic of the clustering pipeline at the bottom shows that the most important change is in disabling the morphological operations (erode and dilate, in transparent gray). As a result, the thresholded binary image contains noisy speckles that are detected as core points in DBSCAN. It can be seen that in the third image (red color code) the clustering fails to properly detect the markers. Instead, a large number of clusters are detected. This is clearly illustrated in the rightmost image, where each point is identified with a numeric label belonging to the cluster ID, where magenta labels represent valid clusters and gray (-1) labels represent outliers or noise. This can be partially remedied by further tuning of the DBSCAN parameters. However, the additional noise in the threshold image will continue to produce problems for the correct detection of the remainder of the markers.

A better approach would be to use DBSCAN clustering differently. In Sec. II.D, a methodology was proposed to approach the DBSCAN clustering from a novel viewpoint. DBSCAN is known for its ability to discard points that are not part of a cluster as noise. Instead of looking for cluster centers, it was proposed to use DBSCAN in an inverse fashion for detection of *noise* (noncore points). In this approach MinPts is set to 1, allowing to maximize the number of points forming a cluster and, instead, to capture the desired clusters by tuning the ϵ and MaxPts parameters. The result of this analysis, with $\varepsilon = 20$, MinPts $= 1$, and MaxPts $= 8$, is shown in Fig. 30. The input to DBSCAN is the same as in the sequence of Fig. 29, except that, as schematized at the bottom of the figure, the inverse DBSCAN filter (cyan block) is applied instead. As a result, the desired clusters (markers) are identified as noise (obtaining a gray -1 label), and the rest of the points are identified as valid clusters, whereas the nominal DBSCAN (Fig. 29) was not able to deal with this without the additional denoising filter. Thus, DBSCAN^{-1} has an advantage over the nominal DBSCAN in this particular scenario.

In essence, the DBSCAN^{-1} approach is actively looking for noise, discarding the actual clusters. Subsequently, the clusters can be retrieved by an additional step where the nominal DBSCAN is applied again. To this end, a new parameter, MaxPts, was introduced, putting a cap on the number of reachable core points within a cluster. Because noise will be randomly and densely cluttered together, the probability is high (for most noise models) that noise particles will be surrounded with a dense number of other noise particles within an arbitrary ϵ neighborhood. It is important to note that this condition holds for MinPts $= 1$, such that the number of points forming a cluster is maximized.

E. Effect of Image Thresholding

Variations of light and motion activity of the object make the task of obtaining a good thresholding challenging. In this study, several thresholding approaches were investigated: global normalization, baseline normalization, and adaptive global thresholding using Otsu's method. The analysis is shown in Fig. 31. Here, the input images from run R1 were converted to grayscale, and thresholding strategy was applied as described in Sec. II.B.3. The corresponding pixel intensity map obtained after thresholding is shown in 3D (left column) and top (right column) view. The regions indicated with dark peaks correspond to high occurrence pixel regions passing the threshold and thus high pixel activity. The scatter points in red correspond to the travel of the marker centers across the image sequence and high wing motion activity. The collected range of input images was arbitrarily chosen at intervals as a continuous vector:

$$N_n = [1 \quad 4 \quad 11 \quad 90 \quad 95 \quad 160 \quad 168 \quad 274 \quad 326 \quad 330 \quad 424 \quad 469]$$

1. *Global Normalization Thresholding*

The results of global normalization are shown in Fig. 31a. Here, the full image sequence was converted to grayscale and normalized in the 0–255 range. Then, a threshold of $\tau_{\text{th}} = 9 \cdot 10^{-5}$ was applied. Subsequently, the pixel intensities across time were summed over the input vector N_n:

$$\sum_{n=0}^{N} G(x,y)_n = \sum_{n=0}^{N} f_{\text{norm}}(I(x,y)_n) \tag{32}$$

A 3D view of the accumulated pixel intensity sum is shown in Fig. 31a. It can be observed that the so-called *footprint* of the

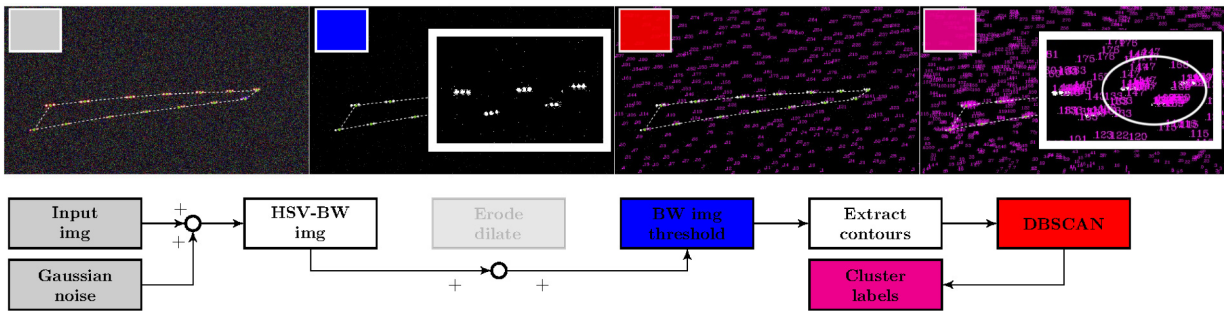


Fig. 29 Tracking sequence on input images (0, 50, 100) from run R1 with injected Gaussian noise ($\mu = 0$ and $\sigma = 0.5$) and disabled morphological operations (transparent block).

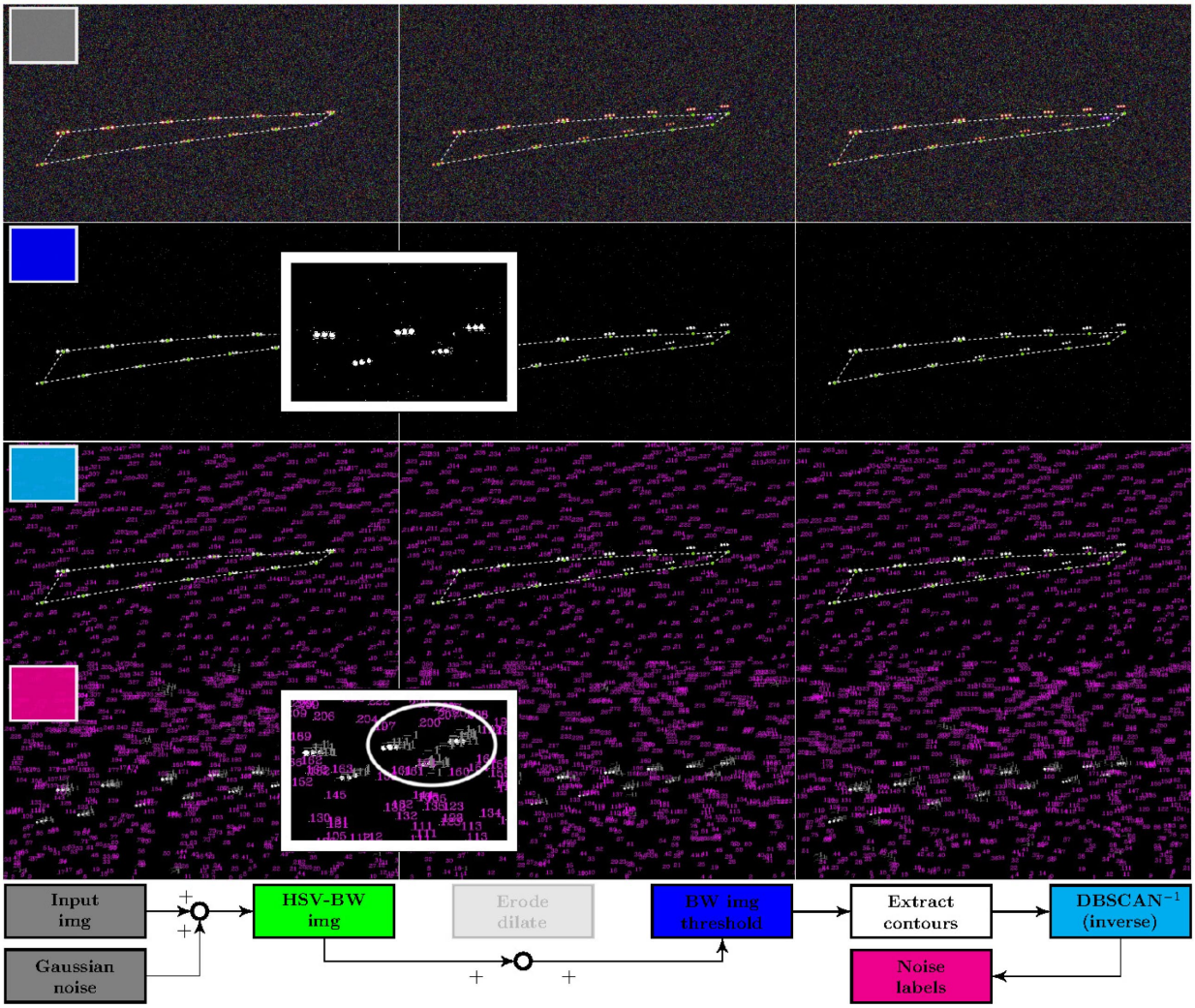


Fig. 30 Tracking sequence on input images (0, 50, 100) from run R1 with injected Gaussian noise ($\mu = 0$ and $\sigma = 0.5$), and DBSCAN^{-1}.

markers, indicated by the red line, corresponds to high pixel activity in the spatial domain across the input sequence. This footprint corresponds to the full sequence from Fig. 21a projected to the spatial domain of the image. The pixel activity is indicated by high-intensity values retained after thresholding (dark peaks). For a continuous input sequence, correct thresholding should characterize the moving foreground as high intensity and filter out the static background. This appears to be the case here; nonetheless, a disbalance in the height of the peaks was observed. The high peaks are observed at the location of the bottom markers (below IDs 14-5 in Fig. 13a), indicating that high-intensity values were captured repeatedly at these locations. Observing the top view plot in Fig. 31b, it can be deduced that the highest motion amplitudes belong to the wingtip, meaning that the outline around the bottom markers should have been more cleanly filtered by the threshold due to the low motion activity.

2. Baseline Thresholding

The results of the baseline thresholding are shown in Figs. 31c and 31d. The main observation of baseline thresholding is that the balance between the peak heights is retained, and the static areas around the bottom markers are correctly filtered. However, because the differences in pixel intensities are now shifted closer together, the background is noisier and distinct static patterns are picked up due to a higher sensitivity to the threshold parameter, which is undesirable. This can be partially remedied by adjusting the threshold parameter.

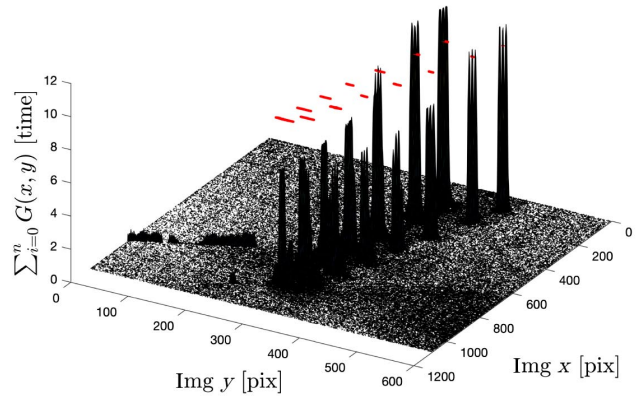
3. Adaptive Otsu's Thresholding

The results of Otsu's thresholding are shown in Figs. 31e and 31f, where a clear definition of a (moving) foreground can be observed. The high pixel intensities are correctly assigned to the marker centers alone, and the background is entirely absent. Figure 31d shows an obvious agreement between the marker footprint and pixel intensity. Also, the height of the peaks is more balanced, meaning that the static areas are efficiently removed. The results indicated that Otsu's method is the cleanest approach for thresholding for this given dataset.

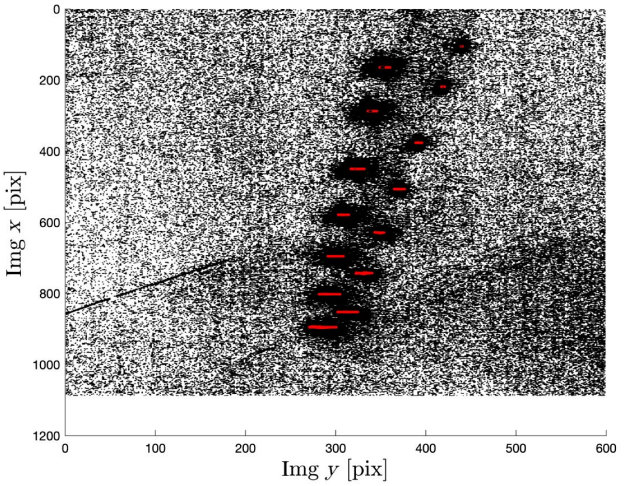
F. Adaptive Thresholding Approach Using Time-Spatial Frequency Content

The methodology explained previously along with the results on the assessment of image thresholding (Figs. 31a, 31c, and 31e), as well as the time traces (Fig. 21a), is indicative of an interconnection between segmentation and clustering. However, the gap is to connect these processes in time, that is, to find a relationship between segmentation and clustering parameters to adaptively obtain an optimal marker label detection through time, for a sequence of images.

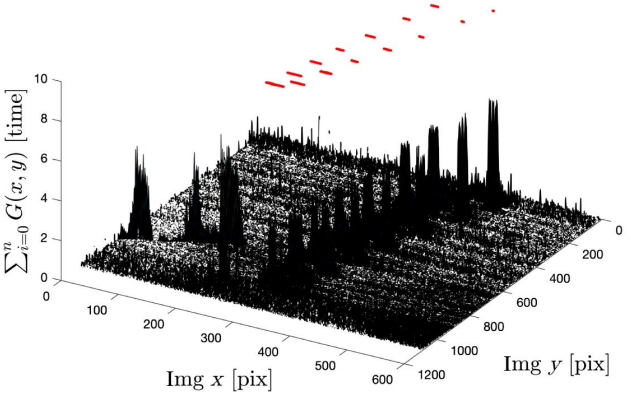
To this end, a method implementing the sliding discrete Fourier transform (SDFT) [49] is suggested. This approach is illustrated in Fig. 32. Here an image sequence is shown with markers indicated in red (not in true scale). A point P_1 corresponding to a marker label with coordinates x_i and y_i, represents a displacement signal in x, y pixel values in time domain. Consequently, its movement corresponds to a *footprint* in the spatial domain. The gray threshold value



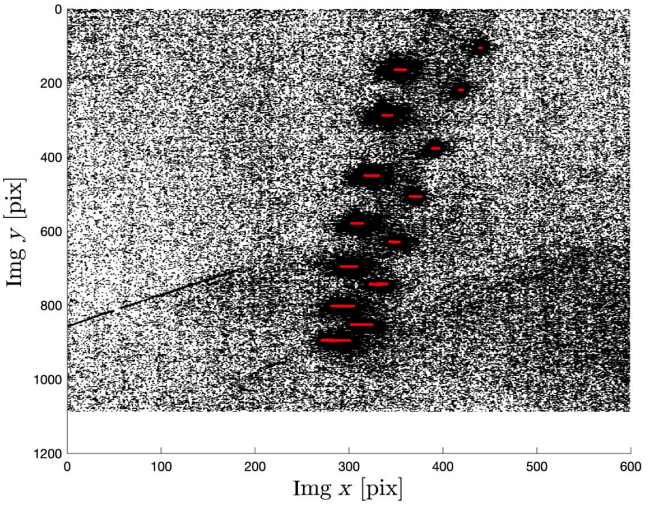
a) 3D view global normalisation thresholding



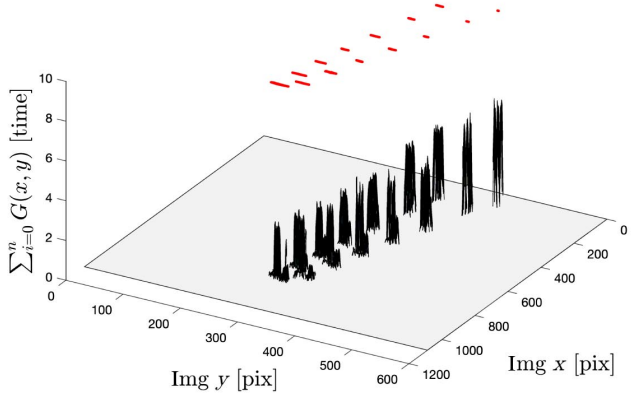
b) Top view global normalisation thresholding



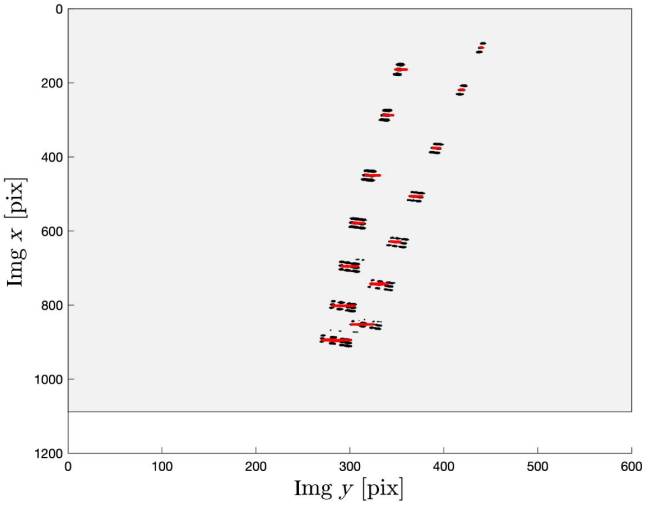
c) 3D view view baseline thresholding



d) Top view baseline thresholding



e) 3D view Otsu thresholding



f) Top view Otsu thresholding

Fig. 31 Three-dimensional (right column) and top (left column) view of the accumulated pixel intensity values after thresholding across the 469 frame sequences of R1.

is adjusted to capture P_1 as it moves in the footprint. As gradually more data reflecting the motion of P_1 are collected in the SDFT time window, the amplitude of the peak starts to become more prominent at a distinct value of the resonance frequency ω_n. This continuously updated knowledge of the wing dynamics projected on the spatial image domain in terms of high pixel activity regions can consequently be used to adjust the parameters of the marker detection pipeline.

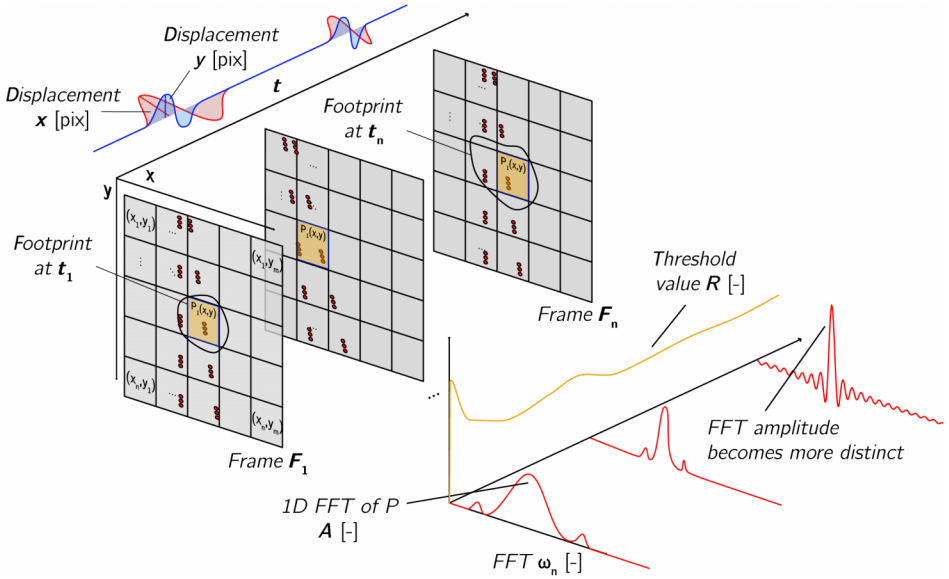


Fig. 32 Simplified spatial and time representation of the SDFT and thresholding adaptation for a sequence of images.

The nature of the SDFT allows for the method to be used in real-time at low cost. Implementing this approach would effectively enable the use of the pipeline as a robust tracking algorithm. In a follow-up study, the role and the benefit of the SDFT are to be further investigated.

V. Conclusions

In this study, an image tracking pipeline was developed using a robust machine learning approach, with the aim to 1) automatically label visual markers and 2) investigate a noninvasive state estimation approach for online control applications of flexible aircraft wings.

The pipeline consisted of an image segmentation, where a mask for clustering operations was obtained with an HSV color filter and a threshold filter using morphological operations (erode and dilate). Subsequently, the mask was clustered using unsupervised clustering with DBSCAN [20]. DBSCAN was compared with another unsupervised clustering method, the disjoint-set data structure [21], by processing 1) a performance data obtained from a performance test with a randomly generated cluster data and 2) experimental data obtained from the measurements of the wing response undergoing oscillatory motion under gust excitations in the OJF wind tunnel at the Delft University of Technology.

The DBSCAN pipeline was found to be more reliable and robust in terms of accuracy and resilience against nose in the input image, which was confirmed with both performance data as well as the experimental data. Next to showing an overall better tracking capability compared with the disjoint-set data structure, an error of $\approx$1 pixel was observed for the majority of the markers clustered with DBSCAN with respect to the validation dataset.

An essential shortcoming of the denoising HSV morphological segmentation filter was highlighted with regard to sensitivity to noise and variations in image illumination. More specifically, 1) the clustering performance degraded without morphological operations and 2) the mask for the clustering operations could be erased by morphological operations under certain lighting conditions (high illumination). This suggested that morphological operations are not always possible or beneficial, and highlighted the relevance and the need for a novel clustering approach.

To tackle this problem, a novel formulation of DBSCAN, the inverse DBSCAN (DBSCAN^{-1}), was proposed, where the clustering problem is reformulated into a noise filtering problem. Instead of rejecting, this approach explicitly detects the noise, making the clustering an implicit task. The experimental dataset was processed using the DBSCAN^{-1} pipeline, and it was shown that the actual clusters were successfully identified and isolated from the noise in the image. After isolation of the clusters, DBSCAN^{-1} must be followed by an additional nominal DBSCAN clustering to extract the exact location of the markers. The final nominal DBSCAN can be done at a significantly lower computational cost due to the removed noise. Further studies are required to assess the performance gain of DBSCAN^{-1} compared with additional filtering steps in various lighting conditions.

In conclusion, the results of the time- and frequency-domain analyses on the experimental data suggested that the motion of an oscillating wing can be adequately captured with relatively low-resolution cameras (1.3 megapixels) in the proposed tracking pipeline. It was suggested that the frequency content of the image sequence could also be extracted by the SDFT and the image thresholding step continuously adapted to produce a better tracking result. The accuracy and motion resolution could be further improved by increasing the resolution of the camera. However, a thorough tradeoff is essential, as this would generally lead to an increase in the computational cost of the tracking pipeline and reduce the maximum processing frame rate, subsequently, reducing the bandwidth of the tracking.

Appendix A: Allegra Wing Specifications

Table A1 shows the specifications of the Allegra wing [38]:

Table A1 Parameters of the Allegra wing

Definition	Parameter	Value	Unit
Span	b	1.6	m
Top chord	c_{root}	0.36	m
Root chord	c_{tip}	0.12	m
Taper ration	λ	1/3	— —
Aspect ratio	A	6.67	m
Sweep (quarter line)	Λ	−17	deg
Wing area	S	0.384	m^2
Mean chord	c_m	0.24	m
Airfoil max. thickness	— —	13.028	%
Airfoil max. camber	— —	2.4	m

Appendix B: Tracking Result for Run R3 with Gaussian Noise

Figure B1 shows the results for the tracking sequence from run R3 with injected Gaussian noise (mean of $\mu = 0$ and standard deviation of $\sigma = 0.5$).

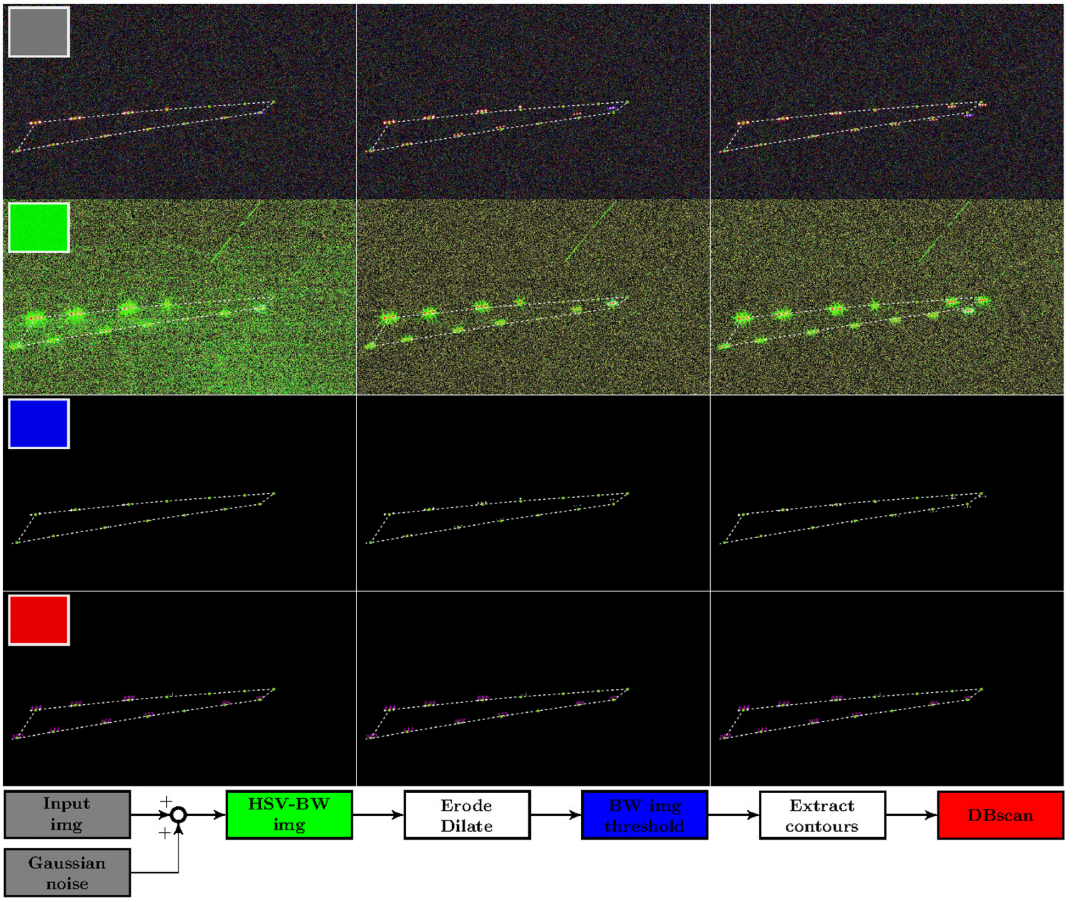


Fig. B1 Tracking sequence on input images (0, 60, 80) from run R3 with injected Gaussian noise ($\mu = 0$ and $\sigma = 0.5$).

Acknowledgments

This research was funded by the Delft University of Technology located in Delft, the Netherlands. The authors would like to thank Johannes Dillinger of the German Aerospace Center (DLR) for providing the wing model and assistance during the experimental data collection. Furthermore, the authors would like to thank the colleagues of the Aerospace Structures and Materials Department for helping to assemble the gust generator.

References

[1] Burner, A. W., and Liu, T., "Videogrammetric Model Deformation Measurement Technique," *Journal of Aircraft*, Vol. 38, No. 4, 2001, pp. 745–754. https://doi.org/10.2514/2.2826

[2] Corke, P. I., "Visual Control of Robot Manipulators—A Review," *Visual Servoing: Real-Time Control of Robot Manipulators Based on Visual Sensory Feedback*, edited by K. Hashimoto, World Scientific, Singapore, 1993, pp. 1–31. https://doi.org/10.1142/9789814503709_0001

[3] Wang, X., "Intelligent Multi-Camera Video Surveillance: A Review," *Pattern Recognition Letters*, Vol. 34, No. 1, 2013, pp. 3–19.

[4] Belbachir, A. N., *Smart Cameras*, Vol. 2, Springer, Boston, 2010. https://doi.org/10.1007/978-1-4419-0953-4

[5] Choi, Y., Martel, M., Briceno, S., and Mavris, D., "Multi-UAV Trajectory Optimization and Deep Learning-Based Imagery Analysis for a UAS-Based Inventory Tracking Solution," *AIAA SciTech 2019 Forum*, AIAA Paper 2019-1569, 2019. https://doi.org/10.2514/6.2019-1569

[6] Hu, Y., Cao, Y., Ding, M., and Zhuang, L., "Airport Detection for Fixed-Wing Unmanned Aerial Vehicle Landing Using a Hierarchical Architecture," *Journal of Aerospace Information Systems*, Vol. 16, No. 6, 2019, pp. 214–223. https://doi.org/10.2514/1.I010615

[7] Abu-Jbara, K., Sundaramorthi, G., and Claudel, C., "Fusing Vision and Inertial Sensors for Robust Runway Detection and Tracking," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 9, 2018, pp. 1929–1946. https://doi.org/10.2514/1.G002898

[8] Agrawa, P., Ratnoo, A., and Ghose, D., "Image Segmentation-Based Unmanned Aerial Vehicle Safe Navigation," *Journal of Aerospace Information Systems*, Vol. 14, No. 7, 2017, pp. 391–410. https://doi.org/10.2514/1.I010457

[9] Valasek, J., Famularo, D., and Marwaha, M., "Fault-Tolerant Adaptive Model Inversion Control for Vision-Based Autonomous Air Refueling," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 6, 2017, pp. 1336–1347. https://doi.org/10.2514/1.G001888

[10] Parsons, C., Paulson, Z., Nykl, S., Dallman, W., Woolley, B. G., and Pecarina, J., "Analysis of Simulated Imagery for Real-Time Vision-Based Automated Aerial Refueling," *Journal of Aerospace Information Systems*, Vol. 16, No. 3, 2019, pp. 77–93. https://doi.org/10.2514/1.I010658

[11] Abousleiman, R. D., Rawashdeh, O. A., and Siadat, M. R., "Statistical Algorithm for Attitude Estimation from Real-Time Aerial Video," *Journal of Aerospace Computing, Information and Communication*, Vol. 7, No. 10, 2010, pp. 322–337. https://doi.org/10.2514/1.47957

[12] Weisshaar, T. A., "Morphing Aircraft Systems: Historical Perspectives and Future Challenges," *Journal of Aircraft*, Vol. 50, No. 2, 2013, pp. 337–353. https://doi.org/10.2514/1.C031456

[13] Cai, L., He, L., Xu, Y., Zhao, Y., and Yang, X., "Multi-Object Detection and Tracking by Stereo Vision," *Pattern Recognition*, Vol. 43, No. 12, 2010, pp. 4028–4041. https://doi.org/10.1016/j.patcog.2010.06.012

[14] Al-Isawi, M. M., and Sasiadek, J. Z., "Control of Flexible Wing UAV Using Stereo Camera," *GeoPlanet: Earth and Planetary Sciences*, Springer-Verlag, Cham, 2019, pp. 97–120. https://doi.org/10.1007/978-3-319-94517-0_7

[15] Mkhoyan, T., de Visser, C. C., and De Breuker, R., "Parallel Real-Time Tracking and 3D Reconstruction with TBB for Intelligent Control and Smart Sensing Framework," *AIAA SciTech 2020 Forum*, AIAA Paper

2020-2252, 2020.
https://doi.org/10.2514/6.2020-2252
[16] Borah, B., and Bhattacharyya, D. K., "An Improved Sampling-Based DBSCAN for Large Spatial Databases," *Proceedings of International Conference on Intelligent Sensing and Information Processing, ICISIP 2004*, IEEE, New York, 2004, pp. 92–96.
https://doi.org/10.1109/icisip.2004.1287631
[17] Smiti, A., and Elouedi, Z., "DBSCAN-GM: An Improved Clustering Method Based on Gaussian Means and DBSCAN Techniques," *INES 2012—IEEE 16th International Conference on Intelligent Engineering Systems, Proceedings*, IEEE, New York, 2012, pp. 573–578.
https://doi.org/10.1109/INES.2012.6249802
[18] Noticewala, M., and Vaghela, D., "MR-IDBSCAN: Efficient Parallel Incremental DBSCAN Algorithm Using MapReduce," *International Journal of Computer Applications*, Vol. 93, No. 4, May 2014, pp. 13–18.
https://doi.org/10.5120/16202-5391
[19] Lu, D., and Weng, Q., "A Survey of Image Classification Methods and Techniques for Improving Classification Performance," *International Journal of Remote Sensing*, Vol. 28, No. 5, 2007, pp. 823–870.
https://doi.org/10.1080/01431160600746456
[20] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X., "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, Vol. 96, AIAA, Reston, VA, 1996, pp. 226–231.
[21] Galler, B. A., and Fisher, M. J., "An Improved Equivalence Algorithm," *Communications of the ACM*, Vol. 7, No. 5, 1964, pp. 301–303, http://portal.acm.org/citation.cfm?doid=364099.364331.
https://doi.org/10.1145/364099.364331.
[22] Sural, S., Qian, G., and Pramanik, S., "Segmentation and Histogram Generation Using the HSV Color Space for Image Retrieval," *IEEE International Conference on Image Processing*, Vol. 2, IEEE, New York, 2002, pp. II-589–II-592.
https://doi.org/10.1109/icip.2002.1040019
[23] Otsu, N., "A Threshold Selection Method from Grey Scale Histogram," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 1, No. 1, 1979, pp. 62–66.
[24] Lancelot, P., Sodja, J., and De Breuker, R., "Investigation of the Unsteady Flow over a Wing Under Gust Excitation," *17th International Forum on Aeroelasticity and Structural Dynamics, IFASD 2017*, IFASD, Como, June 2017, pp. 1–13, https://repository.tudelft.nl/islandora/object/uuid%3A2caf1e73-3de2-4067-803d-8a52c9050fea?collection=research.
[25] Bovik, A., *Handbook of Image and Video Processing*, Academic Press, New York, 2005, pp. III-97–III-309.
https://doi.org/10.1016/B978-0-12-119792-6.X5062-1
[26] Cai, M., Song, J., and Lyu, M. R., "A New Approach for Video Text Detection," *IEEE International Conference on Image Processing*, Vol. 1, IEEE, New York, 2002, pp. I-117–I-120.
https://doi.org/10.1109/icip.2002.1037973
[27] Suzuki, S., and Be, K. A., "Topological Structural Analysis of Digitized Binary Images by Border Following," *Computer Vision, Graphics and Image Processing*, Vol. 30, No. 1, 1985, pp. 32–46.
https://doi.org/10.1016/0734-189X(85)90016-7
[28] Acharya, R., *Multidimensional Image Analysis and Mathematical Morphology*, IEEE, New York, 1989, pp. 203–216.
https://doi.org/10.1109/mdsp.1989.97120
[29] Pal, N. R., and Pal, S. K., "A Review on Image Segmentation Techniques," *Pattern Recognition*, Vol. 26, No. 9, 1993, pp. 1277–1294.
https://doi.org/10.1016/0031-3203(93)90135-J
[30] Liu, J., Li, W., and Tian, Y., "Automatic Thresholding of Gray-Level Pictures Using Two-Dimension Otsu Method," *1991 International Conference on Circuits and Systems*, IEEE, New York, 2002, pp. 325–327.
https://doi.org/10.1109/ciccas.1991.184351
[31] Vala, M., and Baxi, A., "A Review on Otsu Image Segmentation Algorithm," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, Vol. 2, No. 2, 2013, pp. 387–389.
[32] Kittler, J., and Illingworth, J., "On Threshold Selection Using Clustering Criteria," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-15, No. 5, 1985, pp. 652–655.
https://doi.org/10.1109/TSMC.1985.6313443
[33] Shah, G. H., "An Improved DBSCAN, a Density Based Clustering Algorithm with Parameter Selection for High Dimensional Data Sets," *3rd Nirma University International Conference on Engineering*, IEEE, New York, 2012, pp. 1–6.
https://doi.org/10.1109/NUICONE.2012.6493211
[34] Jarvis, R. A., "On the Identification of the Convex Hull of a Finite Set of Points in the Plane," *Information Processing Letters*, Vol. 2, No. 1, 1973, pp. 18–21.
https://doi.org/10.1016/0020-0190(73)90020-3
[35] Mkhoyan, T., de Visser, C. C., and De Breuker, R., "Adaptive Real-Time Clustering Method for Dynamic Visual Tracking of Very Flexible Wings," *AIAA SciTech 2020 Forum*, AIAA Paper 2020-2250, 2020.
https://doi.org/10.2514/6.2020-2250
[36] Jongkind, K., Falkmann, A., and van der Veer, H., "Open Jet Facility," 2020, https://www.tudelft.nl/lr/organisatie/afdelingen/aerodynamics-wind-energy-flight-performance-and-propulsion/facilities/low-speed-wind-tunnels/open-jet-facility/.
[37] "Polytec Single-Point Vibrometers," Polytec, 2020, https://www.polytec.com/us/vibrometry/products/$single-point-vibrometers/.
[38] Ritter, M., Meddaikar, Y. M., and Dillinger, J. K., "Static and Dynamic Aeroelastic Validation of a Flexible Forward Swept Composite Wing," *58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, AIAA Paper 2017-0637, 2017.
https://doi.org/10.2514/6.2017-0637
[39] "Basler ace acA1300-30gm—Area Scan Camera," Basler, 2019, https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca1300-30gm/.
[40] "Computar Machine Vision Lens Catalog," Computar Tech. Rept., 2017, https://computar.com/resources/files_v2/1551/Computar_FAv0817.pdf.
[41] "Embedded Computer Vision Real Time? Nvidia Jetson TX2 + VisionWorks Toolkit—Myzhar's MyzharBot and More . . . ," Myzhar, 2020, https://www.myzhar.com/blog/embedded-computer-vision-real-time-nvidia-jetson-tx2-visionworks-toolkit/.
[42] Bradski, G., "The OpenCV Library," *Dr Dobbs Journal of Software Tools*, Vol. 25, 2000, pp. 120–125.
https://doi.org/10.1111/0023-8333.50.s1.10
[43] "Jetson TX2 GStreamer Guide—Google Search," *NVIDIA*, 2020, https://www.google.com/search?q=Jetson+TX2+GStreamer+guide&oq=Jetson+TX2+GStreamer+guide&aqs=chrome.69i57j0.392j0j4&sourceid=chrome&ie=UTF-8.
[44] "GStreamer: Open Source Multimedia Framework," GStreamer, 2020, https://gstreamer.freedesktop.org/.
[45] Mkhoyan, T., "tmkhoyan/cvyamlParser: Initial Public Release," Zenodo, Nov. 2019.
https://doi.org/10.5281/zenodo.2703498
[46] Mkhoyan, T., "tmkhoyan/adaptiveClusteringTracker: Initial Public Release," Zenodo, 2019.
https://doi.org/10.5281/zenodo.3561015
[47] Mkhoyan, T., "tmkhoyan/autoLabelTool: Initial Public Release," Zenodo, May 2020.
https://doi.org/10.5281/zenodo.3820854
[48] Mkhoyan, T., "tmkhoyan/adaptivePerformanceTest: Initial Release," Zenodo, May 2020.
https://doi.org/10.5281/zenodo.3820916
[49] Jacobsen, E., and Lyons, R., "The Sliding DFT," *IEEE Signal Processing Magazine*, Vol. 20, No. 2, 2003, pp. 74–80.
https://doi.org/10.1109/MSP.2003.1184347

E. Atkins
Associate Editor