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

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

DOI
10.2514/6.2020-2250

Publication date
2020

Document Version
Final published version

Published in
AIAA Scitech 2020 Forum

AIAA SciTech Forum
6-10 January 2020, Orlando, FL
AIAA Scitech 2020 Forum

10.2514/6.2020-2250

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

T. Mkhoyan*, C. de Visser†, R. De Breuker‡
Delft University of Technology, Delft, The Netherlands

Advancements in intelligent aircraft controller design, paired with increasingly flexible and efficient aircraft concepts, create the need for the development of novel (smart) adaptive sensing suitable for aeroelastic state estimation. In contrast to rigid states, aeroelastic state estimation requires more measurement points (displacements and forces) across the span to capture the vibrational shapes of the wing undergoing excitations. A potentially universal and non-invasive approach is visual tracking. However, many tracking methods require manual selection of initial marker locations as the start of a tracking sequence. This study is part of a larger study in the field of smart sensing and aims to cover the gap by investigating a robust machine learning approach for unsupervised automatic labelling of visual markers. The method utilizes fast DBSCAN and adaptive image segmentation pipeline with HSV colour filter to extract and label the marker centres under the presence of marker failure. A comparison is made with Disjoint-set data structure for clustering of the data. The segmentation-clustering pipeline with DBSCAN shows the capability to act as a visual tracking method on its own, capable of running real-time at 250 fps on an image sequence of a single camera with a resolution of 1088×600 pixels. To increase the robustness against noise, a novel formulation of DBSCAN better suited against noise, the inverse DBSCAN ($DBSCAN^{-1}$), is proposed, allowing to cast the clustering problem into noise filtering problem with an additional $MaxPts$ parameter. Furthermore, observations are made regarding the frequency content of the image pixel intensities across time, and how this can be utilized to estimate the natural frequency of the system and adjust the segmentation-clustering pipeline with a sliding DFT (Discrete Fourier Transform).

Nomenclature

$B(x', y')$	= Kernel matrix	$O(...)$	= Computational complexity
D	= Dataset	$P(x,\ y)$	= Density distribution of particles (2D)
$f(I(x, y))$	= Filtering (sequence) operation	$p,\ q$	= Scatter points
$f_{dilate}(I(x, y))$	= Dilate operation	$p_n,\ q_n$	= Scatter noise particles
$f_{erode}(I(x, y))$	= Erode operation	V_∞	= Wind tunnel flow velocity
f_g	= Gust vane frequency	$w_1,\ w_2$	= Class variance weights (Otsu)
$f_{morph}(I(x, y))$	= Morphological operations (combined)	$Z_\epsilon(p_n)$	= ϵ neighbourhood of noise points p_n
f_{norm}	= Global normalisation operation	α_g	= Gust vane angle
$G_f(x, y)$	= Filtered image	ϵ	= Radius of neighbouring points
$I(x, y)$	= Input image	μ_I	= Mean of points in 2D image
$MaxPts$	= DBSCAN$^-$1 maximum points dense region	σ_I	= Standard deviation of points in 2D image
$MinPts$	= DBSCAN minimum points dense region	$\sigma_w^2(\tau_{th})$	= Intra-class variance (Otsu)
$N_\epsilon(p)$	= ϵ neighbourhood of points p	$\sigma_1^2,\ \sigma_2^2$	= Class variances (Otsu)
$N(x,\ y,\ t)$	= Random seed initialised noise mask	τ_{th}	= Threshold parameter

*Ph.D. student, Faculty of Aerospace Engineering, Aerospace Structures and Materials department, T.Mkhoyan@tudelft.nl, P.O. Box 5058, 2600GB Delft, The Netherlands.
†Assistant Professor, Faculty of Aerospace Engineering, Control and Operations department, C.C.deVisser@tudelft.nl, P.O. Box 5058, 2600GB Delft, The Netherlands
‡Associate Professor, Faculty of Aerospace Engineering, Aerospace Structures & Computational Mechanics, R.DeBreuker@tudelft.nl, P.O. Box 5058, 2600GB Delft, The Netherlands

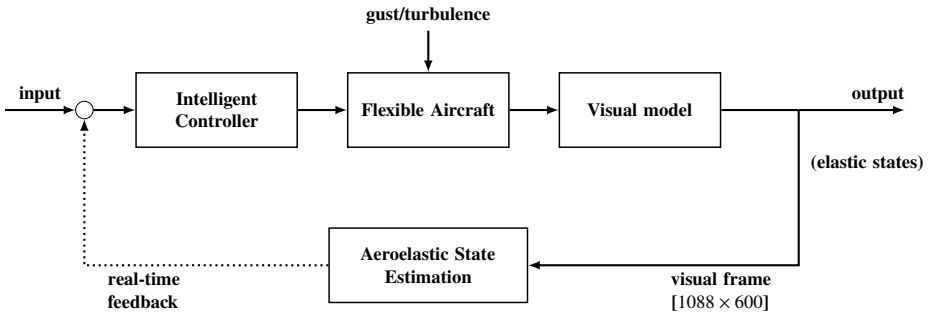


Figure 1 State estimation setup consisting of an intelligent controller, aircraft model, visual model and aeroelastic state estimation using visual tracking. This study aims to contribute to the advancement in state estimation with the dotted feedback loop.

In the context of aeroservoelastic control, monitoring the entire wingspan can be crucial for proper delegation of control actions. This may involve installing many conventional accelerometers that are likely subject to noise and bias, lack of space, certification requirements and challenges associated with correct geometric placement. A smart sensing approach is desired for these types of wing structures, that relies on novel types of sensors to provide feedback to an intelligent controller.

A solution that can greatly reduce the complexity associated with hardware installation, and provide the flexibility needed for using novel state estimation methods, is aeroelastic state estimation by visual methods. An illustration of aeroelastic state estimation using visual data is shown in Fig. 1. Using visual information for observing deformations has been successfully applied on wind tunnel models in early studies [1], and has also seen wide application in robot manipulation [2]. However, in recent years, the capability in terms of on-board computation and camera quality has immensely increased, while the hardware has become more compact [3, 4]. This opens the door for numerous embedded applications using a camera as a sensor. In particular, fuselage-mounted camera systems can provide great advantages for flexible aircraft systems, save costs associated with installation, certification, and have the potential of being non-invasive and universally applicable. More importantly, image data is a rich source of information; collected over a period of time, it unlocks the opportunity to approach the state estimation from a new perspective using machine learning methods. While many suitable tracking methods exist for marker detection, correctly labelling the initial markers in the visual frame is still not a trivial task [5].

In this study, two machine learning methods are implemented for unsupervised clustering of marker labels, meaning that they that 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 is made between the two machine learning methods, DBSCAN [6] and Disjoint-set data structure [7], and segmentation-clustering pipeline is developed based on Hue-Saturation-Value (HSV) [8] and adaptive thresholding with Otsu's method [9]. A novel approach to DBSCAN is proposed, inverse DBSCAN ($DBSCAN^{-1}$), were the clustering problem is reformulated into noise filtering problem and an additional $MaxPts$ parameter is introduced in 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, while the *true noise* in the data is identified explicitly and removed from the dataset in a follow-up step.

To investigate the robustness of the method, the input images are subjected to Gaussian noise and both the nominal DBSCAN as well as DBSCAN^{-1} are assessed in performance with less noise filtering. It was observed that the proposed method is capable of real-time tracking and achieving speeds of 250+ fps (frames-per-second), measured on image sequence of a single camera with a resolution of 1088×600 pixels in a laboratory environment on standard Dell Optiplex 7400 and 2.3 GHz Intel Core i5 16G MacBook. This makes the method suitable for on-line control applications. The approach was tested on an image sequence of a flexible wing equipped with LED (light-emitting diode) markers, undergoing oscillatory motion under gust excitations in the Open Jet Facility (OJF) wind tunnel of the Delft University of Technology. In this experiment, the same gust generator was used as the one developed for OJF in a previous study [10]. Furthermore, the effect of the frequency content is studied to investigate how this can be implemented in the pipeline for adjusting the segmentation and clustering pipeline.

I. Methodology

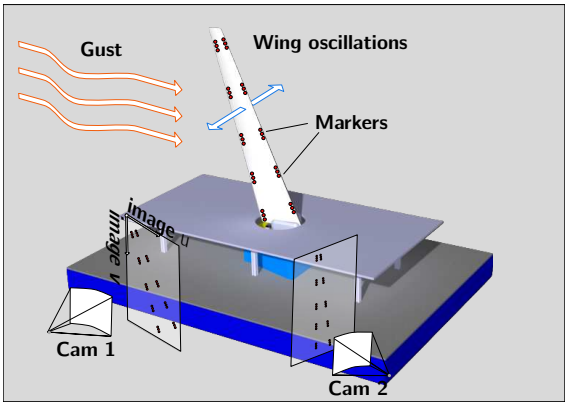


Figure 2 Experimental setup with the wing facing the wind tunnel, equipped with visual markers.

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 [9]. Two machine learning routines are then evaluated: (clustering) DBSCAN [6] and Disjoint-set data structure [7]. 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 image sequence of a flexible wing undergoing motion, equipped with active LED markers.

A. Segmentation

Segmentation approaches are generally focussed 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:

$$G_f(x, y) = f(I(x, y))$$

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

For a sequence of images, the process is a function of the number of frames and thus, implicitly, time [11]. When the desired segments of the image contain colour information, a commonly applied technique is colour filtering in the HSV space. The main benefit of processing in this colour space is that the image intensity and colour can be distinctly separated. The method also has a wide use in video sequence processing and image extraction [8, 12].

1. HSV Filter

To separate the background from the markers, a HSV filtering pipeline is used, composed of multiple filters. First, the image is segmented based on the colour temperature of distinct LED markers, based on distinct values of hue, saturation and value, as shown in Fig. 2. The filter is tuned to find the near optimal HSV values to minimize the noise in the image. In Fig. 3 the result is show of such an operation.

The figures, from left to right, show how the original image is filtered based on its HSV values, obtaining a Black-and-White (binary) (BW) colour-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 [13], where a border following technique is applied with the aid of topological analysis of the contours of a border shape.

In Fig. 3, 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,

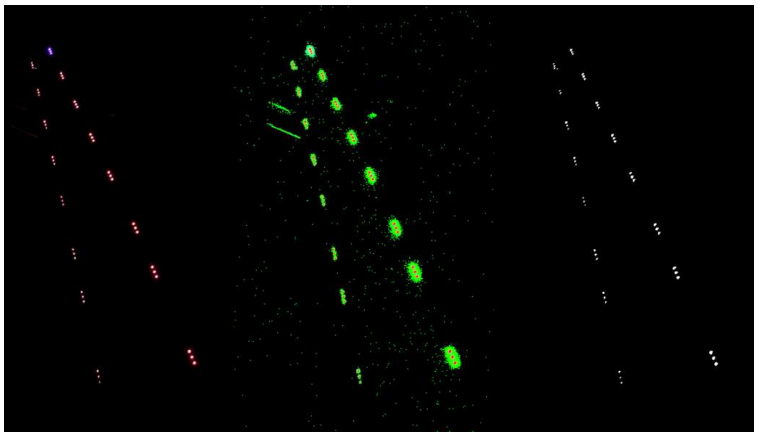


Figure 3 Result of a single HSV filtering operation. The images are from left to right: original, HSV, Black-and-White (binary) (BW) threshold image after colour filtering.

additional scattered background pixels are present in the HSV (middle) image. Since this image is close to bimodal by nature, in this study 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. 4, a simplified schematic is shown of the HSV segmentation and clustering pipeline.

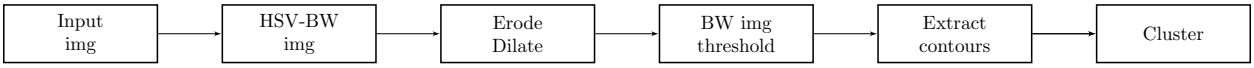


Figure 4 Schematic of HSV filtering, such that a BW image is obtained for subsequent clustering.

2. Morphological operations

HSV filter alone may produce a noisy speckle masked image. A typical way to deal with this is by means of morphological image transformations [14]. Morphological operation are, in general, useful, not only for removal of global noise (e.g. Gaussian noise), but also for isolating and joining separate individual elements. 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 centre. 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:

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

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

and combined operation:

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

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 Section III.

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 [15]. 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, 3 approaches are investigated: using global unit normalisation, baseline normalisation, adaptive global thresholding using Otsu's method [9].

Global normalisation thresholding The global normalisation can be applied by converting the 3-channel RGB input image to greyscale. Subsequently, the image can be scaled with the maximum value of the greyscale, depending on how the greyscale 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:

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

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

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

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

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).

Baseline thresholding In this approach, the baseline pixel intensities are taken into account of the n^{th} 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 similar to Eq. 5, but now the normalisation of n^{th} sequential image is done using:

$$(I(x, y)_n)_{norm} = \frac{I(x, y)_n - \mu_{I_n}}{\sigma_{I_n}} \frac{1}{I(x, y)_{n=0}} \tag{7}$$

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.

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 [9, 16, 17]. It is well suited for images that have a bimodal grey pixel intensity histogram. In the latter 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 inter-class variance is minimized, which would suggest placing the threshold value in the middle of the peaks. The minimisation to find a threshold value of τ_{th} can be represented as:

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

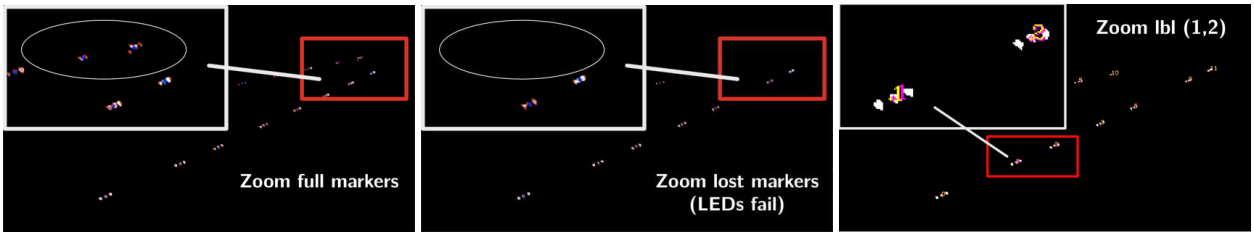
where the parameters w_1, w_2 and σ_1^2, σ_2^2 correspond to the probability and the variance of the two classes and can be computed from the histograms [9].

The limitation of this method is the bimodality assumption, which may not hold for each image and its greyscale image pair [18]. 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 which are capable of dealing with noisy images [16], however, in this regard HSV filtering is responsible for filtering out most of the image noise, making the thresholding less complicated.

B. 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 [6] and the Disjoint-set data structure [7]. These algorithms were particularly suitable due to their unsupervised nature, namely (i) minimum needed domain knowledge, (ii) ability to find clusters of varying size and (iii) ability to deal with noise (in case of DBSCAN). The latter algorithm 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 ($O(n \log(n))$). 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 an unsupervised clustering method. The reason was the periodic failure of led markers installed on the wing. This is illustrated in Fig. 5. The experiment was designed to include robustness, and a condition was tested were under high gust loads and wing oscillations, LEDs started to fail (going on and off), effectively losing certain markers whereby the number of markers (and thus cluster centres) varied over time ánd across experimental runs.



(a) Clustering of a complete set of markers.

(b) Clustering of an incomplete marker set (LEDs are lost), but unsupervised clustering is capable of finding the correct number of clusters.

(c) Clustering result from DBSCAN scan (purple), and the Disjoint-set data structure (yellow); the zoomed region shows marker labels 1, 2 (lbl (1,2)).

Figure 5 Difference in mask obtained after thresholding of an incomplete (left) versus full (middle) set of markers (due to periodic failure of LEDs), showing the necessity of unsupervised clustering. The red dots are contours of the mask and blue is the centroid. The image on the right shows the result of clustering.

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. The latter two parameters are the main parameters of the algorithm. Given a database D, the ϵ neighbourhood, N_ϵ, of point p w.r.t. point q has the following form [6]:

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

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 neighbourhood of border points has much fewer points than the ϵ neighbourhood 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:

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

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

These conditions, thus, set a requirement for every point p in a cluster to be in the ϵ neighbourhood of another point q in this cluster. Additionally, the ϵ neighbourhood 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) [6].

In Fig. 5c the result is shown of the clustering operations both for DBSCAN scan (purple) and Disjoint-set data structure (yellow).

C. Inverse DBSCAN: DBSCAN^{-1} for noise detection of sparse datasets

While 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 to the data. Noise can have various sources, e.g. interference in hardware signal, poor illumination or, simply, poor pre-filtering and thresholding of the input image. In particular for sparse datasets, under such conditions, DBSCAN is known to fail to identify the desired clusters [19]. This arises from the fact that, for a high density of scattered noise, the probability increases such that these noise particles will 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 ($DBSCAN^{-1}$). 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 utilise this formulation of DBSCAN as a noise removal filter, then apply nominal DBSCAN again on clean image domain. To enable this approach, redefinition of DBSCAN is needed and an additional parameter $MaxPts$ is introduced. For a given database D, the ϵ neighbourhood of noise particles p_n and q_n is defined as:

$$Z_\epsilon\left(p_n\right) = q_n \in |dist\left(p_n, q_n\right) \tag{12}$$

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

$$p_n \in Z_\epsilon\left(q_n\right) \tag{13}$$

$$MaxPts \geq |Z_\epsilon\left(p_n\right)| \geq MinPts \quad \text{(core noise particle condition)} \tag{14}$$

To allow this, two conditions must be placed on the DBSCAN^{-1}: (i) $MinPts$ must be set to 1 to capture individual noise particles, (ii) 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 (iii) $MaxPts$ must count *less* points than ϵ neighbourhood of desired cluster points, $N_\epsilon\left(q\right)$. The latter is directly related to the standard deviation, $\sigma_{cluster}$ ($\sigma_{x_{cluster}}$ and $\sigma_{y_{cluster}}$), of $(x,\ y)$ coordinates of a dense cluster and can be chosen based on a priori analysis of the input data set. These conditions dictate that point noise particle p_n does not belong to the ϵ neighbourhood of true clusters N_ϵ, but to Z_ϵ:

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

Density Reachability parameter constraints

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

$$P(x, y)_n < P(x, y)_{cluster} \tag{15}$$

This is, however, a reasonable condition as otherwise, the clustering will not make sense 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 max $MaxPts$ conditions) and reject them as noise. The actual noise particles will meet the core noise particle condition of DBSCAN^{-1} as they do not have a distinct concentrated distribution. A visual representation of this process can be found in the results section.

II. Experimental Setup and Data Collection

The experimental data was collected from camera observations of a flexible wing undergoing gust excitations equipped with active (LED) markers. The study was set up as a larger study on smart sensing methods for control of flexible aircraft.

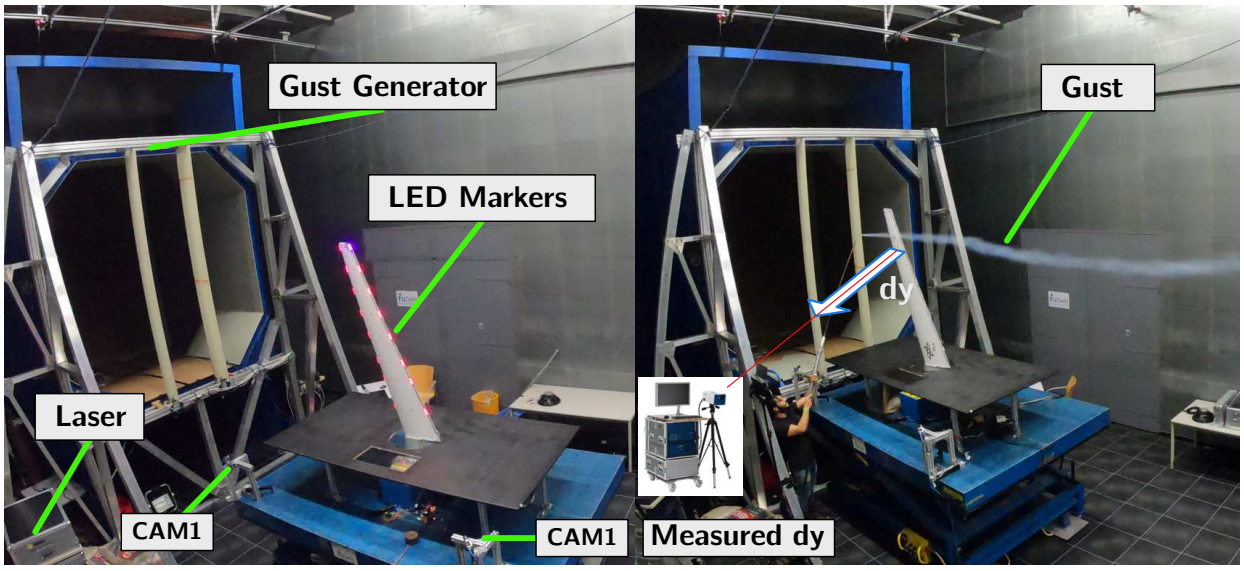


Figure 6 The Open Jet Facility (OJF) located at the Delft University of Technology [20]; the gust generator is mounted in front of the test section.

A. Apparatus

The experiment was conducted in the Open Jet Facility (OJF) at the Delft University of Technology [20]. The OJF, as shown in Fig. 6, 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.

To introduce a gust for the dynamic motion conditions, a gust generator is installed in the test section composed of two foam wings. The gust generator is servo-controlled and allows actuation of the gust vanes at 5-7 Hz for gust vane angle of α_g <= ±15°, and 10-15 Hz for vane angles α_g <= ±10°. The gust generator can produce both harmonic, as well as sinusoid signals of varying frequency. A Polytech PSV-500 laser vibrometer system [21] 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. It was configured to measure 8 markers as shown in Fig. 7a, from a total of 16 LED markers placed on the wing. Since the laser allowed for measurement of only a single point for each run, each run would be repeated 8 times to reconstruct the displacement field of the wing. The system was configured for a sampling rate of 400 Hz.

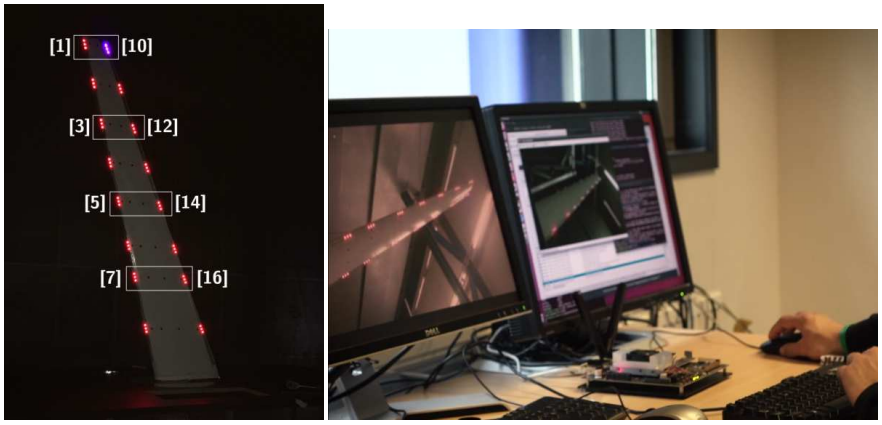
1. Wing model and motion conditions

The wing used in the experiment was a forward swept tapered wing, built of glass fibre reinforced epoxy material, referred to as the *Allegra wing*. The design of the wing allows for large tip displacements, up to 20% for 10° of Angle of Attack (AoA) and 50 m/s flow velocity [22]. 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 a 3 sub-LED units, providing 3 distinct bright light sources per marker. In the experiment a 1-cosine gust signal and a frequency sweep signal were used. The data collected for this particular study contained the following runs, as shown in Table 1:

Table 1 Flow and motion conditions in the wind tunnel collected for the image dataset in terms of number of images and gusts observed. The flow velocity V_{inf} of the wind tunnel, gust vane frequency f_g and gust vane angle α_g. In R3 a sweep signal was applied and no discrete gust.

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	5	inf	30	574	-



(a) View of the wing and the active marker locations tracked by the Polytech measurement system. **(b) Hardware setup for data collection and live tracking with a Jetson TX2**

Figure 7 Experimental Setup

The motion conditions were selected such that it produced a high dynamic response from the wing and thus more 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. Each recoding run R1 and R2 spans observations from 3 gust inputs. The 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. Run R3 was selected since it was designed to show marker loss (LEDs on/off) under high dynamic activity.

2. Dataset collection

The overview of the hardware used for dataset collection is shown in Fig. 7b. The dataset was recorded with two GigE acA1300-75gc ethernet Basler cameras with 1300 CMOS 1.3 megapixel (280×1024 pixels) sensor [23]. The cameras were equipped with Computar 12 mm F1.4 2/3" P IRIS lenses [24] and were positioned in a stereo setup to observe the markers from two viewpoints. The resulting image was cropped to 1088×600 pixel and streamed in 3 channel RGB format synchronously via real-time 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 Frames Per Second (FPS).

The processing power and image capture was 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 [25]. 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 Basler C++ Pylon API [23] and OpenCV open-source computer vision library [26]. The image and tracking data were extracted and plotted in using the OpenCV-Matlab parsing interface tmkhoyan/cvyamlParser [27].

Code development, testing and assessment was done using standard Dell Optiplex 7400 and 2.3 GHz Intel Core i5 16G MacBook and the Jetson TX2. The code, dataset and tolls developed are available under tmkhoyan/adaptiveClusteringTracker [28].

3. Noise model

To evaluate the real-life performance of the method a common noise model is used. The input images are injected with an image independent Gaussian noise and the robustness of the colour filtering, thresholding and clustering pipeline is investigated against possible sensor noise, transmission and hardware related issues and poor illumination. Then, the tracking quality of the pipeline is accessed on image sequences from R1 and R3, where R2 is used as a reference. The probability density function of the Gaussian noise model is as follows:

$$I(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(z-\mu)^2}{2\sigma^2}} \tag{16}$$

In this model, z represents the greyscale value. The parameters used for the noise model are a mean of $\mu = 0$ and a standard deviation of $\sigma = 0.5$. The grey values produced from the probability distribution are scaled to RGB range (0-255) and injected in the 3-channels of the input image $I(x, y)$ producing additive sum of new noise input image $J(x, y)$. The random seed is initialized with the CPU clock for each image input, thus resulting in dynamic noisy image input sequence at each n^{th} frame:

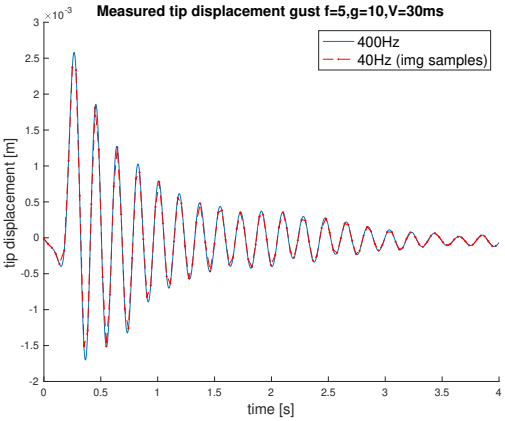
$$I_{noise}(x, y)_n = I(x, y)_n + N(x, y, t) \tag{17}$$

III. Results and Discussion

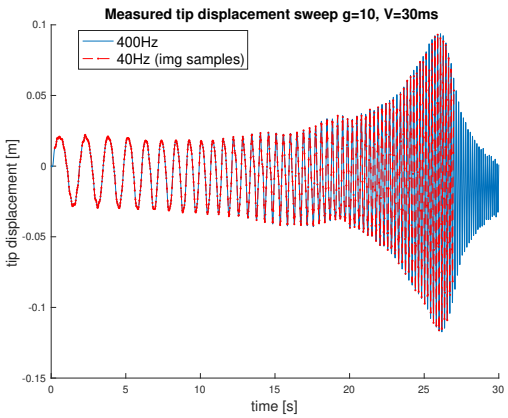
The tracking result with the full clustering pipeline on run R1 and R3 were performed on a sequence of ≈ 468 images from camera 1 (Leading edge). R2 was used as the control for R1 and showed a similar result. R3 was mainly used to asses the ability of the tracking pipeline to deal with marker loss. In the nominal runs, $\epsilon = 20$ pix and reachability parameter of $MinPts = 2$ were used. This set of parameters provided the best cluster detection considering preceding segmentation filters. Speeds of 250+ fps were measured on image sequence of a single camera with a resolution of 1088×600 pixels using standard Dell Optiplex 7400, 2.3 GHz Intel Core i5 16G MacBook and the Jetson TX2.

A. Measured wing response

The laser vibrometer measurement results from the experimental runs R1 and R3 are collected in Figures 8. These results are an indication of the physical motion behind the image sequence and are subsequently compared to tracking results. From the plots, R2 shows a similar response hence it is not included in the set. The time history signals correspond to marker location 1 (as shown in Fig. 7a). Figure 8a show the response of the wing to a single gust input; Fig. 8b shows the response to a sweep signal. The blue curve corresponds to the 400 Hz measurement by the laser vibrometer, the red line is a spline model of this response sampled at the capture intervals of CAM1 (leading edge).



(a) Measurement with 1-cos gust signal: gust vane angle $\alpha_g = 10°$, gust frequency $f_g = 5$ Hz and flow velocity $V = 30$ m/s.



(b) Measurement with sweep signal: gust vane angle $\alpha_g = 10°$, gust frequency $f_g = \text{inf}$ Hz and flow velocity $V = 30$ m/s.

Figure 8 Spline model of the laser measured (400 Hz) tip displacement of marker ID[1] sampled at capture intervals of Cam1 (≈ 40 Hz). From runs with discrete gust and sweep signals.

Looking at the results from Fig. 9, the tracking pipeline shows the ability to follow the motion of the wing and correctly cluster the markers. This is also confirmed by looking at the output of the tracking in terms of pixel (x, y) locations in the images, shown in Fig. 10a. The plots show the time traces for the detected markers and 3 occurrences of decaying sinusoidal responses can be observed from the output. During the image sequence exactly 3 gusts were introduced to the wing that produced a measurement as shown in Fig. 8 for a single gust. It suggests, thus, that the tracking can be used to observe the motion of the wing. Looking at R3 results in Fig. 9, in a bottom row it can be seen how some markers are lost, but DBSCAN is able to correctly cluster the markers without supervision in terms of a number of clusters.

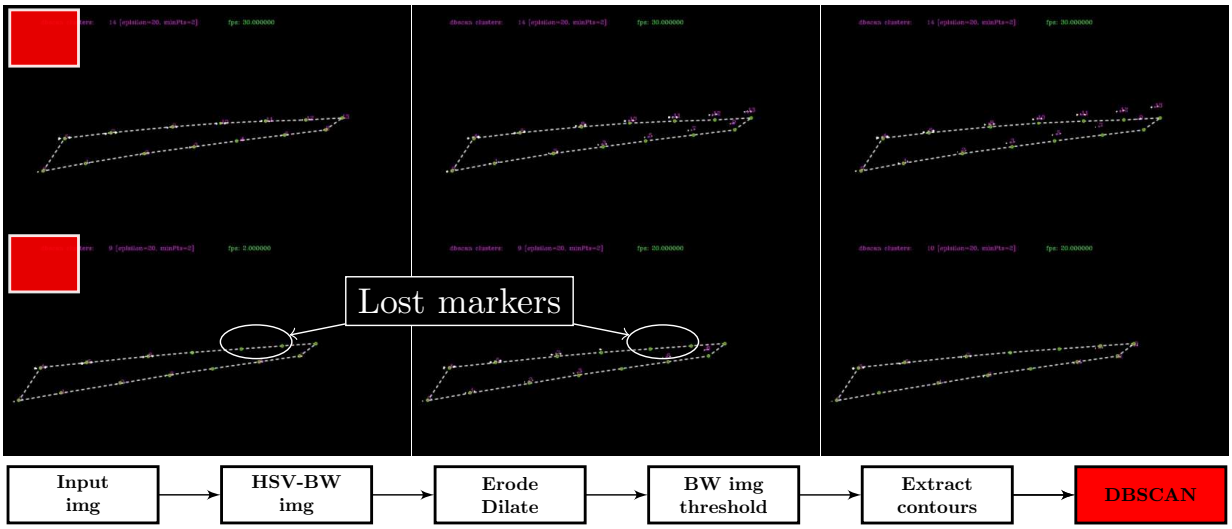
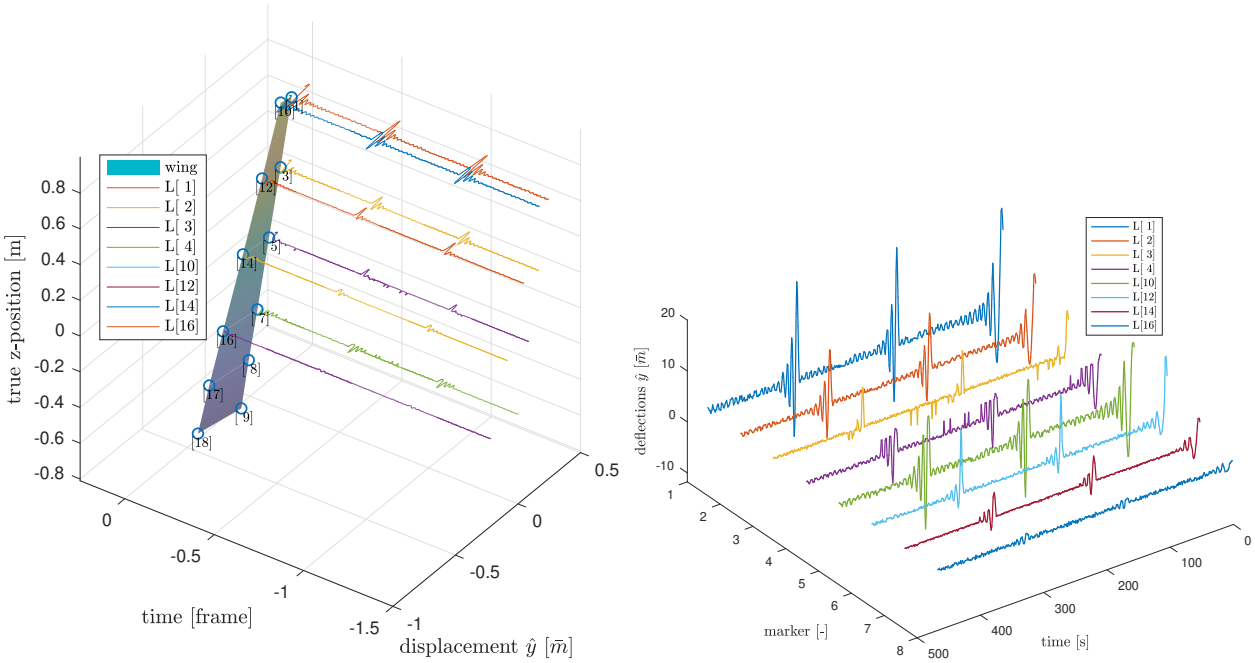


Figure 9 **Tracking sequence on input images from run R1 (upper row) and R3 lower row. The dotted outline shows the initial contour at baseline deflected shape before the gusts hit the wing. The tracking pipeline used is shown in the last row.**



(a) Combined 3D view and time history with marker deflections in image y coordinates.

(b) Relative deflections in y image coordinates with baseline substracted.

Figure 10 **Time series of marker-y deflections in the image coordinates across 498 image sequences, as a result of gust excitations from run R1.**

B. Assessment of DBSCAN parameters

Figure 11 shows the sensitivity of the $MaxPts$ parameter on the tracking result. When the parameter is set to 3, dictating that the Direct Density Reachability of core points needs to contain a neighbourhood of at least 3 core points, the markers 1 and 10 fail to meet these criteria and are no longer score 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 3 sub-LED units can be clotted together in one or two dots instead of 3, hence never meeting the core points condition. It is clearly seen how the cluster is found again once the units become more distinct. It can be seen that this is not happening when the $MaxPts$ parameter is chosen as 2, as shown in Fig. 9, for the reasons explained above.

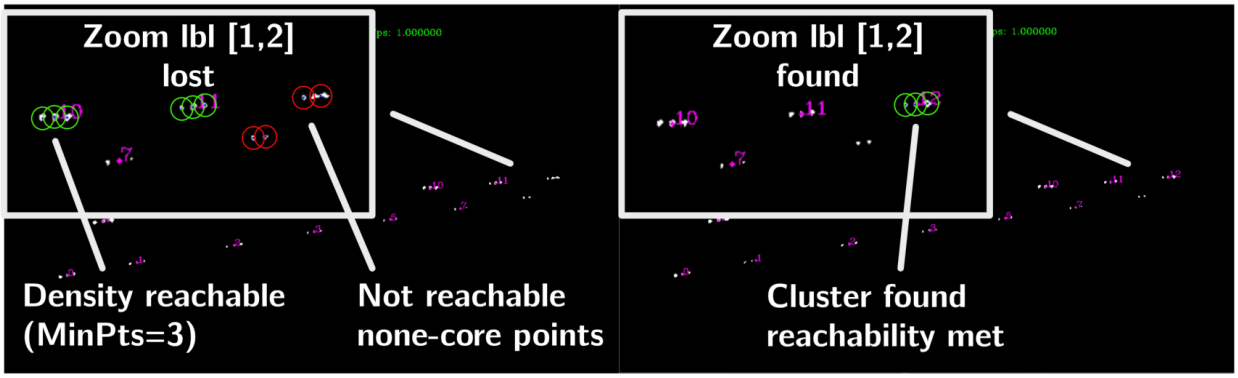


Figure 11 Sensitivity of DBSCAN parameters. Snapshot of two frames from sequence R1, the frames are ≈ 0.025 seconds apart. The parameters are: $\epsilon = 20,\ MinPts = 3$.

C. Evaluation of robustness against noise

The runs R1 and R3 were injected with Gaussian noise ($\mu = 0$ and standard deviation of $\sigma = 0.5$), as explained in the previous section, and the performance of the tracking was evaluated. In Fig. 12 a sequence is shown for tracking of frames 0, 50 and 100. The tracking pipeline used is shown in the bottom row. The colour codes correspond to the stepwise operation performed in the pipeline, throughout the sequence (e.g. green row is HSV filtering operation).

As 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 de-noising filter passing a clean output 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. The results are of the same quality as shown in Fig. 9. For the run R3, a similar result was obtained and can be observed in Appendix B, Fig. 17.

D. DBSCAN^{-1} in the presence of noise without morphological operations

In this analysis, the robustness of DBSCAN was investigated without additional de-noising 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. The latter step showed how well it was able to filter out the remnants of Gaussian noise after HSV operation (row 3), marked as the blue filtering block in Fig. 12. It is therefore interesting to examine what happens when the morphological filter block is removed. The result with the nominal DBSCAN and same parameters ($\epsilon = 20$ pix, $MinPts = 2$) is shown in Fig. 13. Looking at the figure, the most important change in the pipeline is disabling the morphological operations (erode and dilate) shown in transparent grey. As a result of this, the thresholded binary image contains noisy speckles which are detected as core points in DBSCAN. It can be seen that in the third column (red colour code) the clustering fails to properly detect the markers. Instead, a large number of clusters is detected. This is clearly illustrated in the rightmost plot where each point is identified with a numeric label belonging to the cluster ID, where magenta labels represent valid clusters and grey (-1) labels — outliers or noise. This can be partly remedied by further tuning of the DBSCAN parameters, however, the additional noise in the threshold image will continue to produce problems for correct detection of the remainder of the markers.

A better approach would be to use DBSCAN clustering differently. In the section I.C 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 centres, it was proposed to use DBSCAN in an inverse

fashion, for detection of *noise* (non-core points). The approach is to put the $MinPts = 1$, allowing to maximize the number of points forming a cluster and tune instead of the ϵ and $MaxPts$ to capture the desired clusters. The result of this analysis with $\epsilon = 20$, $MinPts = 1$ and $MaxPts = 8$ is shown in Fig. 14. The input to DBSCAN is the same, except, as seen in the last row, the inverse DBSCAN filter (Cyan block) is applied. As a result, as shown in the last row, the desired clusters (markers) are identified as noise (obtaining a grey -1 ID) and the rest of the points are identified as valid clusters, while the nominal DBSCAN (Fig. 13) was not able to deal with this without the additional de-noising filter. Thus, DBSCAN^{-1} has an advantage over the nominal DBSCAN in this scenario.

In essence, the DBSCAN^{-1} approach is actively looking for noise and discards the clusters, subsequently, the clusters can be retrieved by an additional step applying nominal DBSCAN again. For this, a new parameter, $MaxPts$, was introduced putting a cap on the number of reachable core points within a cluster. Since noise will be randomly and densely scattered 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 neighbourhood. It is important to note that this condition will hold when the $MinPts = 1$, such that the number of points forming a cluster is maximized.

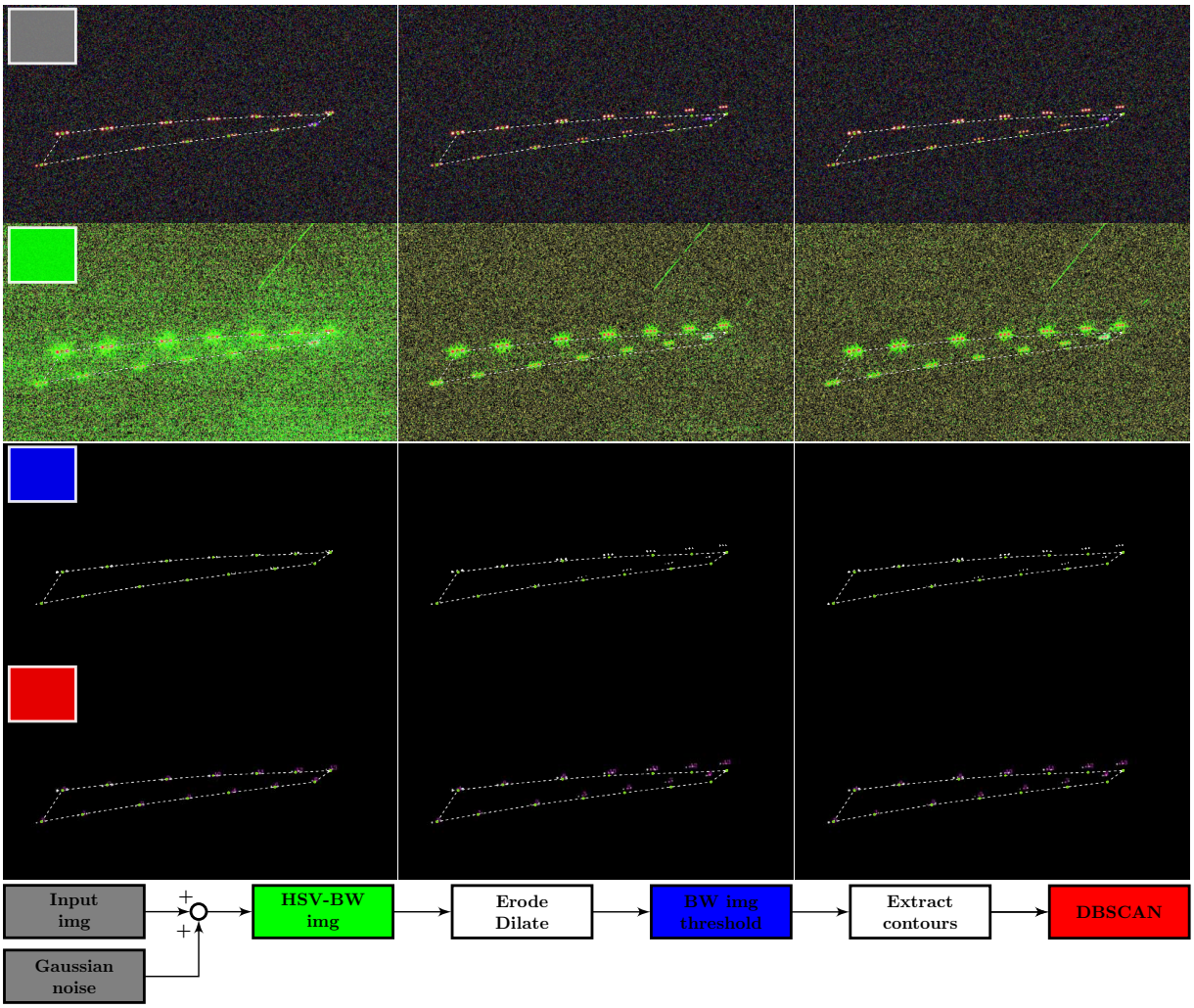


Figure 12 Tracking sequence on input images (0,50,100) from run R1 with injected Gaussian noise ($\mu = 0$ and standard deviation of $\sigma = 0.5$). From top to bottom, the rows represent: input+noise image (gray), HSV filtering (green), threshold image (blue) and clustering result (red). The colour codes correspond to the tracking pipeline (last row) and the dotted outline shows the initial contour at baseline deflected shape before the gusts hit the wing.

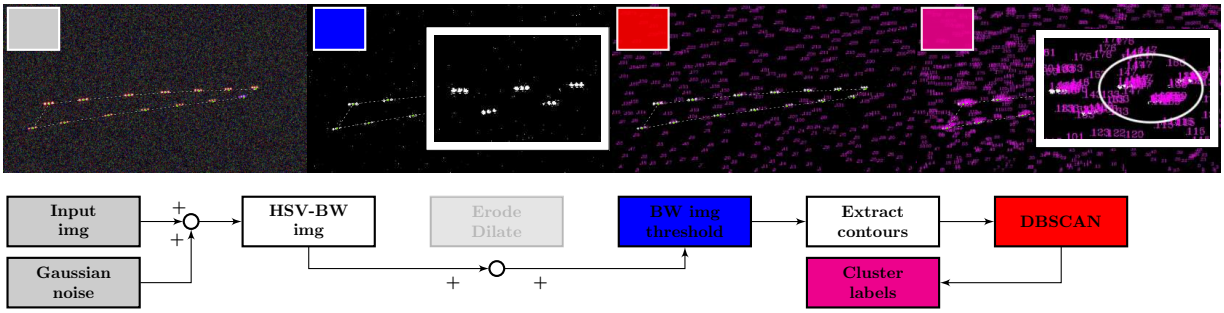


Figure 13 **Tracking sequence on input images (0,50,100) from run R1 with injected Gaussian noise ($\mu = 0$ and standard deviation of $\sigma = 0.5$) and disabled morphological operations (transparent block). From left to right: noisy input image, thresholding after HSV and skipping erode and dilate operations, clustering result and the extracted labels of all points processed in DBSCAN. The modified pipeline is shown in the bottom row.**

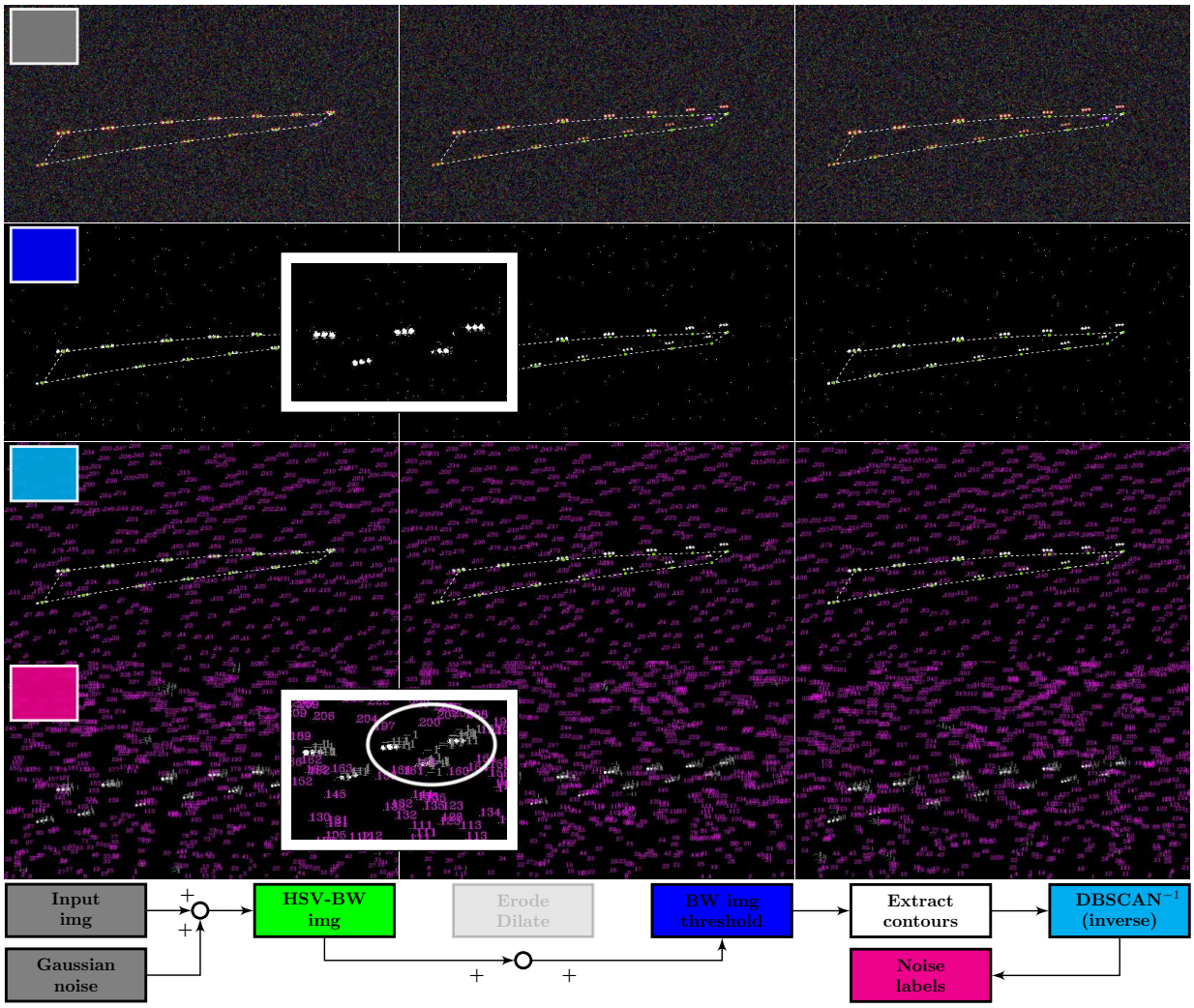


Figure 14 **Tracking sequence on input images (0,50,100) from run R1 with injected Gaussian noise ($\mu = 0$ and standard deviation of $\sigma = 0.5$), and DBSCAN^{-1}. From top to bottom, the rows represent: input+noise image (grey), HSV filtering (green), threshold image (blue) and clustering result (red). The colour codes correspond to the tracking pipeline shown in the last row. The dotted outline shows the initial contour at baseline deflected shape before the gusts hit the wing.**

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 Figures 15a, 15c and 15e. Here, the input images from run R1 were converted to greyscale and thresholding strategy was applied as described in section I.A.3. The collected range of input images was arbitrary chosen at intervals as a continuous vector:

$$N_n = \begin{bmatrix} 1 & 4 & 11 & 90 & 95 & 160 & 168 & 274 & 326 & 330 & 424 & 469 \end{bmatrix}$$

1. Global normalisation thresholding

The result of global normalisation is shown in Fig. 15a. Here, the full image sequence is converted to greyscale and normalized in the $0 - 255$ range. Then, a threshold of $\tau_{th} = 9 \cdot 10^{-5}$ is applied. Subsequently, the pixel intensities across time were summed over the input vector N_n:

$$\sum_{n=0}^{N} G(x, y)_n = \sum_{n=0}^{N} f_{norm}(I(x, y)_n) \tag{18}$$

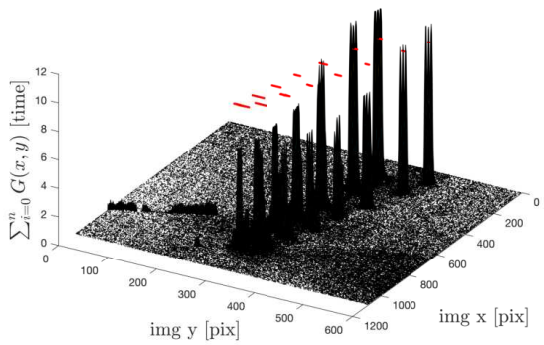
3D view of the accumulated pixel intensity sum is shown in Fig. 15a. It can be observed that the so-called *footprint* of the markers, indicated by the red line, corresponds to high pixel activity in the spatial domain across the input sequence. This footprint corresponds to the full sequence from Fig. 10a 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 assign the moving foreground as high intensity and filter out the static background. This appears to be the case here, however, we see a dis-balance in the height of the peaks. The high peaks at the location of the bottom markers on the wing (below ID's 14-5 in Fig. 7a) indicating that high-intensity values were captured repeatedly at these locations. Observing the top view plot in Fig. 15b it can be seen that the highest motion amplitudes belong to the wing tip, meaning the outline around the bottom markers should have been more cleanly filtered by the threshold due to the low motion activity.

2. Baseline thresholding

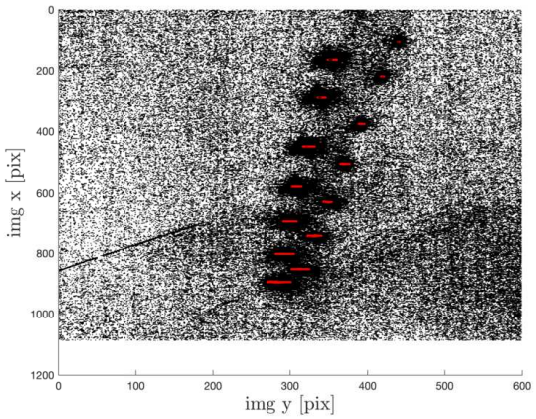
The results of the baseline thresholding is shown in figures 15c and 15d.The main observation of baseline thresholding is that the balance between the peak heights is retained, and the static areas around the bottom markers are correctly filtered. However, since now the pixel intensities differences are shifted closer together, the background is more noisy and distinct static patterns are picked up due to higher sensitivity to the threshold parameter which is undesired. Partly this can be remedied by adjusting the threshold parameter.

3. Adaptive Otsu thresholding

The results of Otsu's thresholding is shown in Figures 15e and 15f. Observing the figures we see a very clear definition of a (moving) foreground. The high pixel intensities are correctly assigned to only the marker centres and the background is entirely absent. Figure 15d shows a clear agreement of marker footprint and pixel intensity. Also, the height of the peaks is more balanced, meaning that the static areas are efficiently removed. It also indicated that this is the cleanest approach for thresholding for this given dataset.

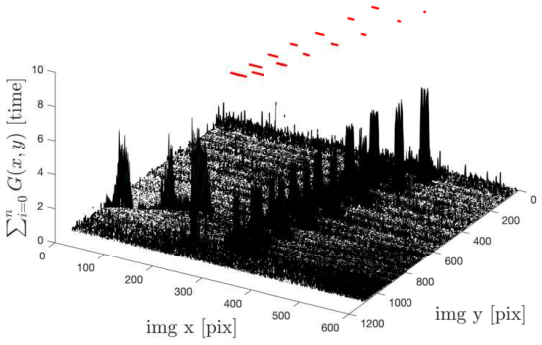


(a) 3D view global normalisation thresholding

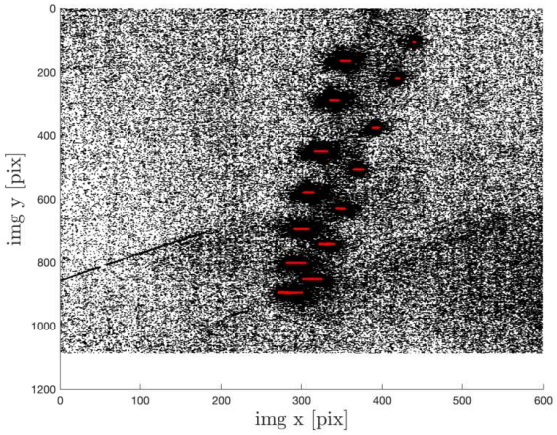


(b) Top view global normalisation thresholding

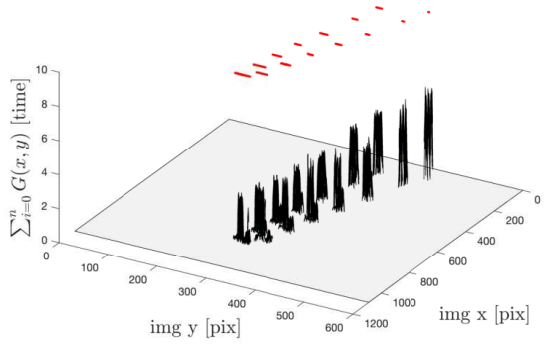
Baseline thresholding



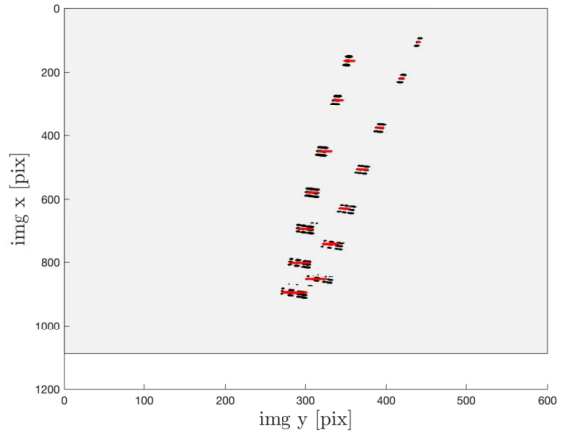
(c) 3D view view baseline thresholding



(d) Top view baseline thresholding



(e) 3D view Otsu thresholding



(f) Top view Otsu thresholding

Figure 15 (Right) 3D view of the accumulated pixel intensity values after thresholding, across the 469 frame sequences of R1. (Right column) Top view of the intensity map with footprints of the markers centres in the spacial image domain representing the spatial image domain in pixels. The regions with high value (dark peaks) correspond high occurrence of respective pixels passing the threshold and thus high pixel activity. The region marked with the red outline corresponds to the travel of the marker centres in the and thus high physical motion of the wing.

F. Segmentation-Clustering pipeline using time-spatial frequency content

The methodology explained previously along with the results on assessment of the image thresholding (Figures 15a, 15c and 15e) as well as the time traces (Fig. 10a), show the interconnection of 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 obtain optimal marker label detection through time for a sequence of images.

For this, a method implementing SDFT (Sliding Discrete Fourier Transform) [29] is suggested. The concept is to monitor and detect regions in the image that show movement (activity) relative to a static scene (background). This would allow one to obtain a footprint of markers in spatial image domain (x, y) through time and adjust the parameters of the image processing (segmentation) and machine learning (clustering) methods more optimally. This is illustrated in Fig. 16 along with additional explanation. The nature of SDFT allows to use it real-time at low cost. Implementing this approach would effectively enable the use of the pipeline as a robust tracking algorithm. In this study, the role and benefit of SDFT are investigated. The methods presented in this study and the processing pipeline were developed in C++ programming language using the OpenCV open-source computer vision library [26].

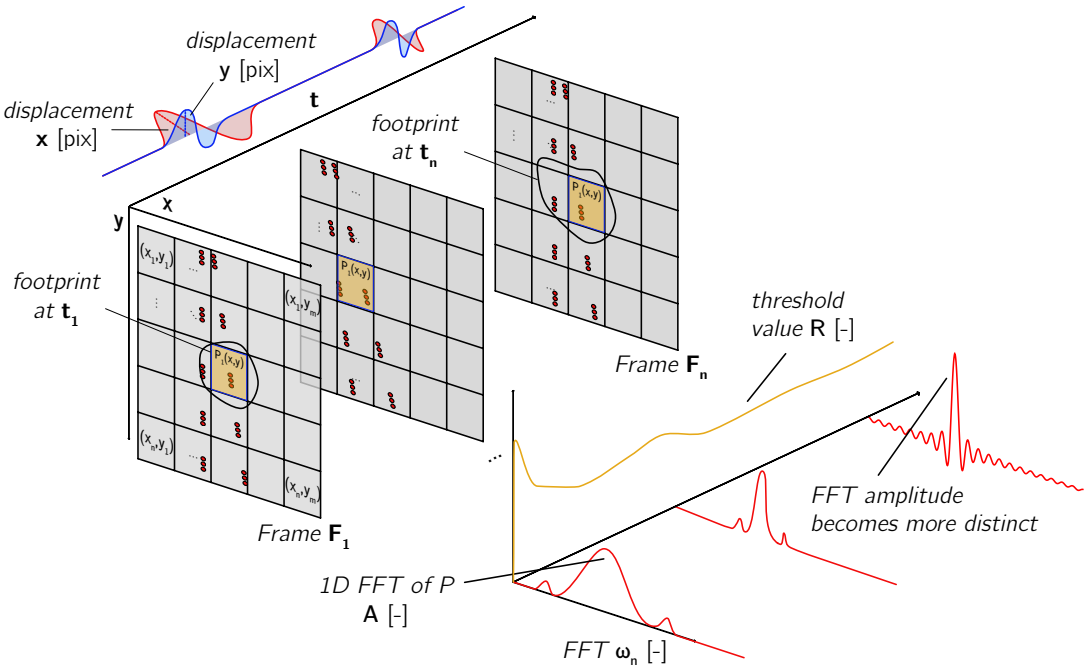


Figure 16 Simplified spatial and time representation of the information extraction process from a sequence of frames n. Note that the image and markers are not represented in true scale. A point P_1 corresponding to a marker label with coordinates x_i and y_i, represents a displacement signal in x, y pixel values in time domain. Consequently, its movement corresponds to a 'footprint' in the spatial domain. The grey threshold value is adjusted to capture P_1 as it moves in the footprint. As the sliding DFT collects more data of the motion of P_1 in x, y the amplitude of the peak starts to become more prominent at a distinct value of frequency ω_n. This gives information about the dynamics of the underlying body in motion and can be used to adjust the parameters of the marker detection pipeline.

IV. Conclusion and Recommendation

In this study, a tracking pipeline was proposed consisting of segmentation and clustering using computer vision and machine learning methodologies. The sequence of images from experimental runs was colour filtered with an HSV colour filter and threshold filter to obtain a mask for clustering operations. The mask was then clustered using unsupervised method DBSCAN [6]. DBSCAN was compared to another unsupervised clustering method, Disjoint-set data structure [7] and found to outperform in terms of computational cost.

It was observed that the method is capable of real-time tracking and speeds of 250+ fps were achieved on an image sequence of a single camera with a resolution of 1088×600 pixels in a laboratory environment on standard Dell Optiplex 7400 and 2.3 GHz Intel Core i5 16G MacBook. This makes the method suitable for on-line control applications. The approach was tested on image sequence of a flexible wing equipped with LED (light-emitting diode) markers, undergoing oscillatory motion under gust excitations in the OJF (Open Jet Facility) wind tunnel of the Delft University of Technology.

The time traces (Fig. 10a) indicated that the method was able to extract dynamic information about the underlying model (wing) under observation. Further experiments and processing are needed to asses the Segmentation-Clustering pipeline as a self-sustained tracking method. Looking at the assessment of the image thresholding (Figures 15a, 15c, 15e.) it was observed that the motion of the markers reflected in the summed pixel intensity map and that the choice of the thresholding influenced the representation of this motion in the spatial domain. The time traces of the footprints throughout the image sequence showed a correlation with the intensity peaks obtained and was a good indication of motion in the spatial domain. The interconnection of thresholding and spatial motion throughout the image sequence indicated that possibly better thresholding can be achieved if an SDFT is applied to extract the frequency of motion along the time axis and the thresholding is continuously adapted.

To investigate the robustness of the method the input images were subjected to zero mean Gaussian noise and both the nominal DBSCAN as well as DBSCAN^{-1} were assessed in performance with reduced noise filtering steps (no morphological operations). The tracking pipeline with HSV filtering and morphological operations (erode and dilate) showed robustness against noise. However, when the latter was turned off for the robustness assessment, the clustering performance degraded. To remedy this, an inverse DBSCAN model, DBSCAN^{-1}, was proposed that reverses the clustering problem and acts as a noise filter. DBSCAN^{-1} must be followed by an additional nominal DBSCAN clustering to extract the true location of the markers. The final nominal DBSCAN can be done at a significantly lower computational cost due to the removed noise. It was shown that with its explicit definitions for noise and core points, the nominal DBSCAN method allowed redefining the model and addressing noise in the data in a novel inverse fashion. Further studies are required to asses the additional complexity of computation and the benefit of DBSCAN^{-1} compared to additional filtering steps.

References

[1] Burner, A. W. and Liu, T., "Videogrammetric model deformation measurement technique," *Journal of Aircraft*, Vol. 38, No. 4, 2001, pp. 745–754.

[2] Corke, P. I., "Visual control of robot manipulators–a review," *Visual Servoing: Real-Time Control of Robot Manipulators Based on Visual Sensory Feedback*, World Scientific, 1993, pp. 1–31.

[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, 2010.

[5] Lu, D. and Weng, Q., "A survey of image classification methods and techniques for improving classification performance," 2007.

[6] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al., "A density-based algorithm for discovering clusters in large spatial databases with noise." *Kdd*, Vol. 96, 1996, pp. 226–231.

[7] Galler, B. A. and Fisher, M. J., "An improved equivalence algorithm," *Communications of the ACM*, Vol. 7, No. 5, 1964, pp. 301–303.

[8] 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, 2002.

[9] Otsu, N., "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, Vol. 9, No. 1, 1979, pp. 62–66.

[10] 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*, 2017.

[11] Bovik, A. C., *Handbook of image and video processing*, Academic press, 2010.

[12] Cai, M., Song, J., and Lyu, M., "Automatic Text Detection and Tracking in Digital Video," *Proceedings. International Conference on Image Processing*, Vol. 1, 2002, pp. I–117–I–120.

[13] Suzuki, S. et al., "Topological structural analysis of digitized binary images by border following," *Computer vision, graphics, and image processing*, Vol. 30, No. 1, 1985, pp. 32–46.

[14] Serra, J., *Image analysis and mathematical morphology*, Academic Press, Inc., 1983.

[15] Pal, N. R. and Pal, S. K., "A review on image segmentation techniques," *Pattern Recognition*, Vol. 26, No. 9, 1993, pp. 1277–1294.

[16] Liu Jianzhuang, Li Wenqing, and Tian Yupeng, "Automatic thresholding of gray-level pictures using two-dimension Otsu method," Institute of Electrical and Electronics Engineers (IEEE), dec 2002, pp. 325–327.

[17] Vala, H. J. 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.

[18] 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.

[19] 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, NUiCONE 2012*, 2012.

[20] TU Delft, "Open Jet Facility," .

[21] Polytec, "Polytec SINGLE-POINT VIBROMETERS," .

[22] Ritter, M., Meddaikar, Y. M., and Dillinger, J. K. S., "Static and Dynamic Aeroelastic Validation of a Flexible Forward Swept Composite Wing," *Scitech*, , No. January, 2017.

[23] Basler, "pylon Linux ARM | Basler," .

[24] Nmatthews, "Computar Machine Vision Lens Catalog," Tech. rep.

[25] NVIDIA, "High Performance AI at the Edge | NVIDIA Jetson TX2," .

[26] Bradski, G., "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[27] Mkhoyan, T., "tmkhoyan/cvyamlParser: Initial public release," .

[28] Mkhoyan, T., "tmkhoyan/adaptiveClusteringTracker: Initial public releas," .

[29] Jacobsen, E. and Lyons, R., "The sliding DFT," *IEEE Signal Processing Magazine*, Vol. 20, No. 2, 2003, pp. 74–80.

Appendix A. Allegra Wing Specifications

Below the specification of the Allegra wing are presented [22]:

Table 2 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. thicknes	–	13.028	[%]
Airfoil max. camber	–	2.4	$[m]$

Appendix B. Tracking result for run R3 with Gaussian noise

Below the result are shown for tracking the sequence from run R3 with injected Gaussian noise ($\mu = 0$ and standard deviation of $\sigma = 0.5$).

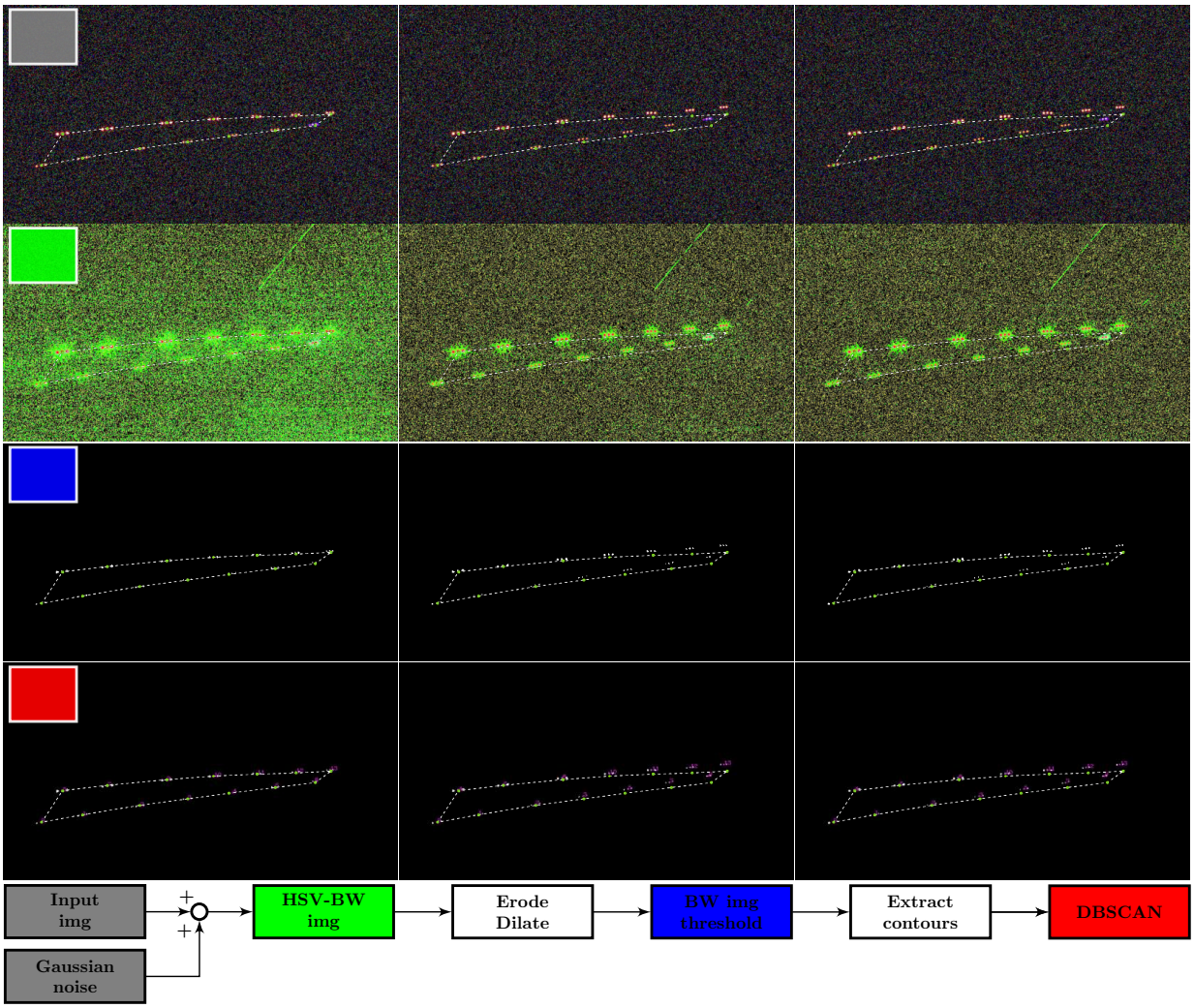


Figure 17 Tracking sequence on input images (0,60,80) from run R3 with injected Gaussian noise ($\mu = 0$ and standard deviation of $\sigma = 0.5$). From top to bottom, the rows represent: input+noise image (grey), HSV filtering (green), threshold image (blue) and clustering result (red). The colour codes correspond to the tracking pipeline shown in the last row. The dotted outline shows the initial contour at baseline deflected shape before the gusts hit the wing.