Stairway to Abstraction: an Iterative Algorithm for Whisker Detection in Video Frames

Jan-Harm L.F. Betting, Vincenzo Romano, Laurens W.J. Bosman, Zaid Al-Ars, Chris I. De Zeeuw, Christos Strydis
Erasmus MC, Rotterdam, The Netherlands

***Abstract*—Automated whisker tracking is important for researching active touch in rodents. Earlier efforts to detect whiskers and represent them in a small set of parameters were either not accurate enough to enable tracking over time, or computationally expensive. In this article we propose an algorithm to cluster whisker centerline points, detected through a curvilinear structure algorithm, using the shape of smaller clusters to form bigger clusters of centerline points. After that, a least-squares approach is used to define each whisker by a set of four parameters. We implemented the algorithm in MATLAB in a parallelized fashion, and found that the processing time per frame is reasonable in MATLAB, and is likely to be short when ported to a lower-level language. When tested on a 33,634-frame segment, 89.2% of the whiskers could be represented in an abstract fashion by four parameters with a mean-squares fitting error of lower than 10 pixels, and visual inspection shows that crossing whiskers are detected and parameterized in an accurate way.**

***Index Terms*— Biology computing, Clustering methods, High performance computing, Image processing, MATLAB, Parallel algorithms, Parallel processing**

I. Introduction

One of the most important ways in which rodents learn about their environment is by active touch: they bring their whiskers in contact with nearby objects. The intricate whisker system of mice is of special interest to scientists: these animals usually live in dark environments and are heavily dependent on active touch. The movement of the whiskers in response to stimuli can give a lot of information about the way these animals learn from external stimuli, and the connection between brain activity and whisker movement. For instance, the method of tracking whisker movement in top-level videos of a mouse has been used to establish that a modifiable, reflexive whisker protection can be evoked by touch [6].

At the Erasmus Medical Center in Rotterdam (The Netherlands), experiments are conducted with head-fixed mice. The whiskers are filmed with a top-view, high-speed camera, recording at frequencies up to 1000 Hz; an example of a frame from such a video is shown in Figure 3A. Detecting individual whiskers on these videos is difficult: they spread, cross, align, and hide behind each other. Furthermore, the camera generates large amounts of frames: even a recording time of a few minutes yields hundreds of thousands of frames, which makes it unfeasible to track the whisker movements manually. The use of software to perform this task is therefore a necessity.

In this paper, we present an algorithm that avoids the need to try all combinations of whisker points, by clustering iteratively. Using a centerline detection technique that can detect whiskers with sub-pixel accuracy, we were able to combine whisker points into small clusters, iteratively merging those clusters based on their shape, eventually defining every whisker as a separate cluster, which can then be represented in an abstract way by parameter fitting.

First, we will summarize the technique that allowed us to detect the centerline of whiskers with sub-pixel precision. We present the iterative clustering technique, and show how the clusters can be used to describe the whiskers in an abstract way. Then, we will show the performance and complexity of a MATLAB executable implementation of this algorithm on actual whisker frames.

II. Related work

When it comes to whisker detection and tracking, a couple of efforts exist. The most comprehensive one so far has been the BIOTACT Whisker Tracking Tool (BWTT), a MATLAB application based on the ViSA algorithm [5]. The authors opted to only detect whisker points in a narrow band around the snout. The algorithm divides pixels in three groups based on their distance to the snout; and tries to connect all the detected pixels from the 'inner' group to those of the 'outer' group, selecting the most feasible options for constructing whisker shafts. This method is computationally expensive, as a large number of combinations of pixels needs to be attempted. Furthermore, the choice to detect only short, linear whisker shafts makes individual whiskers almost impossible to distinguish in video fragments.

Clack et al. used a local-minima algorithm to detect whisker pixels, and a parameterized line detector to find the sub-pixel position of the whisker points. When whiskers cross, their method leaves a gap, which is filled up by interpolation [1]. When we tested their method on our videos, we found that the algorithm has difficulties when trying to accurately describe crossing whiskers; they are often detected as two separate curves, which leads to tracking errors in the video.

Knutsen et al. use a convolution-based algorithm to track rat whiskers by manually locating whiskers on the first frame, and then fitting a piecewise polynomial function. This worked well for trimmed rats, but required intervention for

978-1-7281-3427-7/20/$31.00 ©2020 IEEE

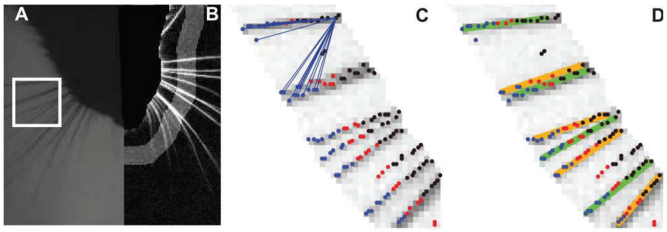

Fig. 1. ViSA algorithm for whisker extraction. Whisker pixels are extracted from a narrow band around the snout. The detected pixels are divided into three groups, and linearized whisker shafts are detected by drawing lines between whisker points from the 'inner' group to those of the 'outer' group. Image taken from [5].

untrimmed rats. As for mice, the algorithm was only validated in untrimmed animals [4].

III. Proposed Algorithm

Whiskers on top-view images can be described as curvilinear structures: their width is negligible compared to their length. The ViSA algorithm uses a local intensity difference detection algorithm to detect which pixels are part of whiskers. Whisker shafts are then detected by a connecting pixels from the outside of the band to those on the inside of the band. The detection algorithm, however, only allows for pixel accuracy. A better way to trace the whiskers is to approach the shape of the section of a whisker by a second-order Taylor polynomial, and finding its vertex analytically, as was done in [7]. In this way, it is possible to use differences in intensity between pixels to estimate the exact position of the centerline, rather than just determining whether a pixel is part of the whisker or not. When we applied this technique to our preprocessed whisker images, we found that the centerlines of the whiskers were detected with barely any noise, as can be seen in Figure 2.

The centerline detection algorithm detects centerline points, but not whiskers. Due to differences in intensity among whiskers on the frames and the limited resolution of the videos, it was not possible to find a configuration for the curvilinear structure detection algorithm that would produce fully continuous centerlines. Furthermore, when whiskers cross, the lower whisker is bound to being partially obscured, which leads to an interrupted centerline.

The clustering and abstraction algorithm we propose makes use of the fact that, as clusters grow, they define the local shape of the whisker. This information can then be used to merge collinear clusters, and eventually even identify partially obscured whiskers.

A. Local clustering

The first step consists of locally clustering individual centerline points. The main information we have here is the position of the whiskers points on the image. Furthermore, the curvilinear structure detection algorithm gives us some information about the local direction of the structure at every point.

We considered two options for local clustering: the DBSCAN algorithm [3], which clusters points based on local distance (if the distance between two points is smaller than a parameter ε, the points are clustered together). We also considered the algorithm described in the Steger paper, where centerline points in neighboring pixels are clustered using a linear combination of distance and difference in local direction as a guideline.

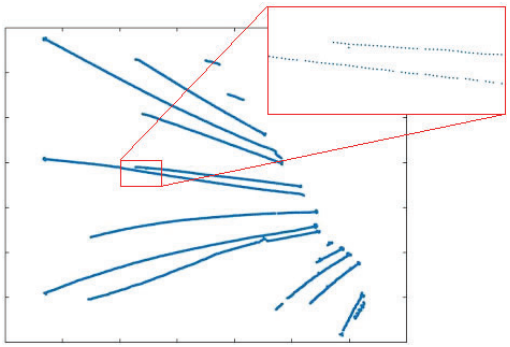

Fig. 2. Whisker centerlines detected using the Steger algorithm. It is clear that there is very little noise. When we zoom in to a place where two whiskers cross, we find that the centerline points are well-aligned, even though the centerline is not fully continuous.

Theoretically, using both location and local direction information would lead to more accurate clustering than using just location information. In practice, we found that the difference in performance for our application between the two algorithms is minor, but it is essential that parameters be well-chosen to prevent incorrect clustering. Because the centerlines are interrupted, the number of clusters will still be higher than the number of whiskers after the local clustering step, as can be seen in Figure 3B.

B. Collinear cluster merging

Once smaller groups of centerline points are formed, it becomes possible to further reduce the number of clusters by merging those that line up (since those are likely to be part of the same whisker). For clusters of a certain size, the direction of a cluster of a certain shape can be determined with help of the covariance matrix of the x- and y-coordinates of the points in the cluster: the eigenvector corresponding to the absolute largest eigenvalue of the covariance matrix corresponds to the direction of the cluster. This eigenvector can be used to construct a rotation matrix to align the cluster along the x-axis.

Since whiskers are curved structures, and even the clusters obtained with local clustering can be long and curved, and the average direction of the cluster can differ from the direction at the tips of the cluster. However, by aligning the cluster along the x-axis, it becomes easy to extract n points from either end of the cluster, in order to create two sub-clusters (the top and the bottom ends). The covariance matrix can again be used to determine the direction of the cluster at the tips.

Knowing the direction and the position of the tips of the clusters, it becomes possible to determine the Euclidean distance between the tops and bottoms of different clusters, and to determine the difference in direction between tops and bottoms. A linear combination of these two parameters can be used to determine how likely it is that two clusters are part of the same whisker (on the condition that the position of the bottom of one cluster relative to the snout is higher than the tip of the other one). In this way, we can assess the tops of all clusters, determine which of the other clusters is collinear (and this part of the same whisker), and merge those clusters. If necessary, this procedure can be repeated, though usually

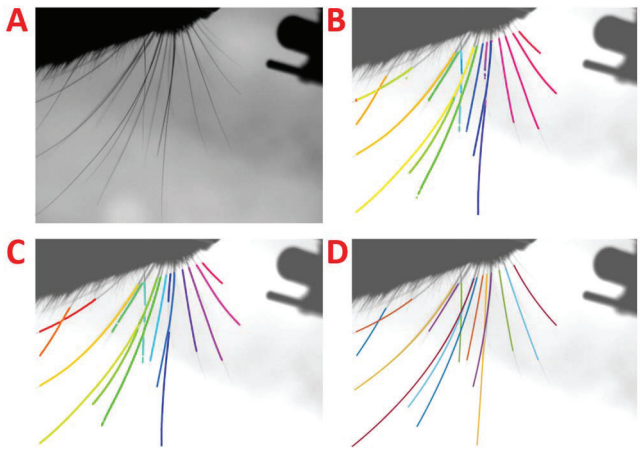

Fig. 3. Frame from a whisker video. (A) shows the original frame. (B) shows the frame after local clustering, and (C) shows it after collinear cluster merging. Different colors denote different clusters. (D) shows a reconstruction of the whiskers based on a set of four parameters per whisker. The last whisker was dropped before parameterization based on its length.

one iteration is enough. Eventually, we obtain one separate cluster of centerline points for each whisker, as in Figure 3C.

C. Parameterization

Once each whisker is represented by a single cluster of centerline points, the whisker needs to be described by a limited number of parameters. In order to follow whiskers over multiple frames, it is important that these parameters fulfil the so-called compactness hypothesis: whiskers that are very different should have very different parameterizations, and whiskers that are similar (or a single whisker on two subsequent video frames) should have similar parameters.

The videos that are used at the Erasmus MC contain free-moving whiskers that do not touch objects. The shape of these whiskers can be described well by a second-order polynomial. Furthermore, the length of a whisker is important, as well as its angle and position relative to the snout (mice are able to shift the pivot point around which whiskers rotate). We can thus describe a single whisker with a set of four parameters: the length L, the position ρ on a line along to the snout, the angle θ relative to that line, and the bending parameter b, which defines the deviation $d=bx^2$ from an axis x, defined by θ and ρ. The parameters can be fit to the cluster of centerline points by means of nonlinear regression, using least-squares estimation. Figure 3D visualizes the parameterized whiskers, and it can be seen that they closely resemble the detected clusters of centerline points shown in Figure 3C.

IV. Implementation and Performance

We implemented the whisker detection algorithm in MATLAB. The preprocessing step, consisting of standard image processing procedures (such as gamma correction) and removal of the background and animal silhouette, were adopted from BWTT. The centerline detection algorithm can be performed in parallel per pixel and was therefore suitable for processing on a GPU. In MATLAB, this can be done by casting the image as a *gpuArray*, which moves it to GPU memory. Per-pixel processing can then be done using the *arrayfun* function.

The collinear cluster merging algorithm was implemented in MATLAB and accelerated by data-parallelizing the assessment of each cluster per iteration, by using a parallel for-loop (*parfor*). Furthermore, the m-code script was compiled into a MATLAB executable (MEX) file to speed up execution. The parameterization algorithm was not compiled, as the built-in nonlinear regression function of MATLAB could not be embedded into a separate MEX file. The collinear cluster merging step is performed a maximum number of three times, but stops as soon as there are no collinear clusters left.

TABLE I
Algorithm Steps

Step name	Parallelizability	Execution time per frame	Avg. nr. of clusters before processing
Local clustering	Per pixel	128.3 ms	5,323
Collinear cluster merging	Per cluster	46.83 ms	73.73
Parameterization	Per whisker	72.26 ms	16.95

Execution time results per step of the process, over a 33634-frame video fragment. Also the parallelizability and the number of clusters before processing is shown (the parameterization step does not reduce the number of clusters unless a minimum length is given as a parameter).

The algorithm was run on a computer with an AMD Ryzen 7 1800X processor @ 3.6 GHz, with 32 GB RAM and a GeForce GTX 1080 Ti GPU, running Windows 10 Enterprise and MATLAB R2018b. We used a video fragment of 33,634 frames. The video had an original resolution of 640x480, and was upscaled by a factor of 2 as part of the preprocessing step. The video, of which one of the frames is visible in Figure 3A, contains a challenging whisker situation: there are multiple crossing and overlapping whiskers. There is also a long hair, which crosses multiple whiskers.

The parallelizability per step, the execution time per frame, and the average number of clusters before processing are shown in Table I. The total execution time for clustering and parameterization is 247.4 ms.

Porting the MATLAB code to a low-level programming language or a different architecture will lead to a further speedup; also, the steps can be implemented as a pipeline, to decrease the processing time per frame.

The quality of the algorithm was tested on two criteria: visual quality and execution time. Execution time was measured in MATLAB using the *tic* and *toc* functions. Visual quality was measured by the squared-error of the parametrization fit: if the error is low, that shows that the cluster is likely to accurately represent a whisker; a high error indicates either incorrect clustering, or a whisker shape that cannot accurately be represented by our bending parameter *b*. We also used visual inspection to see whether the reconstructed image of whiskers accurately

TABLE II
Algorithm Steps

Whisker	MSE of fit
1	0.0239
2	0.1238
3	1.0280
4	6.0417
5	3.6123
6	0.0274
7	0.0316
8	0.4537
9	0.0024
10	3.1131
11	107.32
12	0.0017
13	0.0739
14	0.3947

MSE of each of the fitted whiskers in Figure 3. The whiskers are numbered by their starting point from left to right.

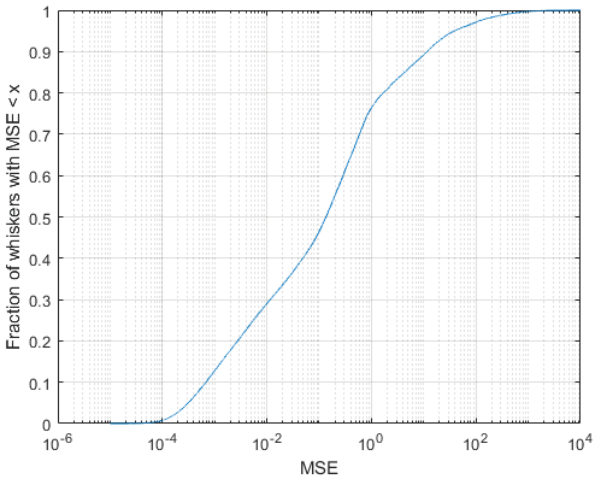

Fig. 4. Fraction of whiskers with an MSE lower than x for a video fragment of 33634 frames. 89.2% of whisker parameterizations has a MSE of lower than 10 pixels.

represents the whisker profile that is visible on the original frames.

Figure 3 shows the results of the different clustering steps, with the original frame in light grayscale shades on the background. The figure shows that the centerlines of the whiskers are detected accurately. The different colors denote different clusters. The long hair is detected as a whisker; this not surprising, since it is a long and moving object, and it is not problematic; it can easily be filtered out afterwards. Future work includes the design and implementation of an algorithm that can track detected whiskers over longer video segments.

Figure 3C shows that the algorithm is able to cluster crossing whiskers correctly. Figure 3D shows that four parameters can accurately represent the original whiskers. There are a few differences: the tips of some of the whiskers on the right are too faint to be detected by the algorithm, which is why their reconstructions are shorter than the originals.

Table II shows the MSE that was calculated after fitting parameters to each of the whiskers in Figure 3. Looking closely at Figure 3D, it can be seen that the whiskers with a high MSE do indeed fit the centerline shape a bit less well, but even whisker 11 (with a MSE of more than 100) fits the shape of the whisker quite accurately. As for the whole video segment, Figure 4 shows that a large majority of the whiskers has a MSE that is low enough to be considered an accurate fit. This shows that the algorithm behaves well over longer segments, too.

V. Conclusion

We designed a three-step algorithm for the detection and parameterization of whiskers in top-view high-speed videos of free-moving whiskers. The algorithm builds on a curvilinear structure centerline detection algorithm, and makes use of the fact that the direction of a cluster can be determined through the covariance matrix of the coordinates of its members. By determining the direction and position of the tops and bottoms of the clusters, it is possible to determine which of the clusters are collinear, which means that the clusters are part of the same whisker. In this way, it is possible to separate the centerline points of different whiskers and make sure that each whisker has its own cluster, even when whiskers are partially obscured.

The whiskers can be represented in an abstract fashion by four parameters that represent physical properties: length, bending, angle and position. When we reconstruct the original image, we see that these parameters can be used to reconstruct the original whisker accurately. We tested the algorithm on a challenging whisker fragment and found that it was able to accurately describe crossing whiskers.

The algorithm performed well in MATLAB, with the total processing time per frame of 247.4 ms. Since the algorithm is highly parallelizable and the different processing steps can be implemented as a pipeline, an implementation in a lower-level language will likely reduce the processing time even further, which makes it useful for processing high amounts of frames from a high-speed camera.

The whisker detection algorithm presented in this paper can be implemented in a larger whisker-tracking software package, which tracks individual whiskers from beginning to end in longer video segments. This would make it possible to study the movements of individual whiskers, instead of the simple average of the detected whiskers, which would make it possible to study the intact whisker system of mice in unprecedented detail.

VI. References

[1] N.G. Clack et al., "Automated Tracking of Whiskers in Videos of Head Fixed Rodents", *PLoS Computational Biology*, vol. 8, no. 7, Jul. 2012.

[2] R. Duin and D. Tax, "Statistical Pattern Recognition," in *Handbook of Pattern Recognition and Computer Vision,* eds. C. Chen and P. Wang, 3rd ed., 2005, pp. 3-24.

[3] M. Ester et al., "A density-based algorithm for discovering clusters in large spatial databases with noise", *Proc. KDD-96*, 1996, pp. 266-231.

[4] P.M. Knutsen, D. Kerdikman, and E. Ahissar, "Tracking Whisker and Head Movements in Unrestrained Behaving Rodents", *Journal of Neurophysiology* 93, 2294-2301, 2005.

[5] I. Perkon et al., "Unsupervised quantification of whisking and head movement in freely moving rodents," *Journal of Neurophysiology* 105, pp. *1950-1962*, 2011.

[6] V. Romano et al., "Adaptation of Whisker Movements Requires Cerebellar Rotentiation," *eLife*, Dec. 2018.

[7] C. Steger., "An unbiased detector of curvilinear structures," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20, pp. *113-125*, 1998.