DefocusTracker

A Modular Toolbox for Defocusing-based, Single-Camera, 3D Particle Tracking

Barnkob, Rune; Rossi, Massimiliano

Published in:
Journal of Open Research Software

Link to article, DOI:
10.5334/jors.351

Publication date:
2021

Document Version
Publisher's PDF, also known as Version of record

Journal of open research software

DefocusTracker: A Modular Toolbox for Defocusing-based, Single-Camera, 3D Particle Tracking

SOFTWARE METAPAPER

RUNE BARNKOB
MASSIMILIANO ROSSI
Author affiliations can be found in the back matter of this article

ubiquity press

ABSTRACT

The need for single-camera 3D particle tracking methods is growing, among others, due to the increasing focus in biomedical research often relying on single-plane microscopy imaging. Defocusing-based methods are ideal for a wide-spread use as they rely on basic microscopy imaging rather than requiring additional non-standard optics. However, a wide-spread use has been limited by the lack of accessible and easy-to-use software. *DefocusTracker* is an open-source toolbox based on the universal principles of General Defocusing Particle Tracking (GDPT) relying solely on a reference look-up table and image recognition to connect a particle's image and its respective out-of-plane depth coordinate. The toolbox is built in a modular fashion, allowing for easy addition of new image recognition methods, while maintaining the same workflow and external user interface. *DefocusTracker* is implemented in MATLAB, while a parallel implementation in Python is in the preparation.

CORRESPONDING AUTHOR:
Rune Barnkob
Heinz-Nixdorf-Chair of Biomedical Electronics, Department of Electrical and Computer Engineering, Technical University of Munich, Center for Translational Cancer Research (TranslaTUM), Munich, Germany
rune@barnkob.com

KEYWORDS:
particle tracking; velocimetry; PTV; general defocusing particle tracking; fluid dynamics; MATLAB; Python

TO CITE THIS ARTICLE:
Barnkob R, Rossi M 2021 *DefocusTracker*: A Modular Toolbox for Defocusing-based, Single-Camera, 3D Particle Tracking. *Journal of Open Research Software*, 9: 22. DOI: *https://doi.org/10.5334/jors.351*

(1) OVERVIEW

INTRODUCTION

The use of single-camera 3D particle tracking analysis is receiving increasing interest, among others, due to the rapid development of bio-engineering and biomedical sciences where single-access imaging, such as with microscopes, is a standard research tool [20, 21]. For this, methods based on the principle of particle image defocusing are particularly attractive as no special optics or cameras are required, and have potential for wide-spread use. However, until now most of the software for defocused-based particle tracking has been developed in-house for private use of research groups, and there are only few examples of user-friendly software that can be accessible to a larger audience, including researchers outside the engineering or computer-science community. One example is GDPTlab, a MATLAB GUI implementation written by the authors and released in 2015 [9]. GDPTlab was used by several research groups and cited in several peer-reviewed journals (see also Reuse potential section). GDPTlab, however, was not distributed under an open-source license, thus its potential for collaboration and expansion was limited by that.

To accommodate this need, we developed *DefocusTracker*, which is a modular and open-source toolbox for defocusing-based 3D particle tracking. *DefocusTracker* uses different architecture and functions and is not compatible with GDPTlab, however they are based on the same method, namely the General Defocusing Particle Tracking (GDPT). GDPT relies on a reference look-up table, or more generally on a training set of labeled data, where defocused particle images are linked with their true depth positions. An image recognition method is trained on this set in order to determine the depth position of target particles based on their defocused images [3, 5, 20]. The GDPT principle is shown in ***Figure 1***. Panel (a) shows the creation and training of a calibration model through calibration/training images of defocused particle images with known 3D particle positions. In Panel (b) the trained model is used to reconstruct the 3D particle positions from the particles' defocused images in 2D measurement images. For more details on the GDPT method, experiments, and uncertainty assessment, we refer to Refs. 2, 3, 5, and 9.

DefocusTracker is built in a modular fashion to allow for a continuous addition of state-of-the-art image recognition methods. The current implemented image recognition method, referred to as normalized_crosscorrelation_3d (or Method 1), is based on the normalized cross-correlation function and described in Ref. 15. Methods based on convolutional neural networks and deep learning are under development but their performance still compares poorly to the cross-correlation approach [2]. Future improved methods based on deep learning will be included in *DefocusTracker*.

DefocusTracker is implemented in MATLAB, while a parallel implementation in Python is in the preparation. *DefocusTracker* is accompanied by a website, *https://defocustracking.com/*, to facilitate the research community with a platform for sharing of user guides, experiences, and applications, as well as for data for training and validation.

IMPLEMENTATION AND ARCHITECTURE

General architecture

The general architecture and workflow is shown in ***Figure 1(c)***. The toolbox is based on three main types of data structures and five primary functions for their creation, processing, and manipulation:

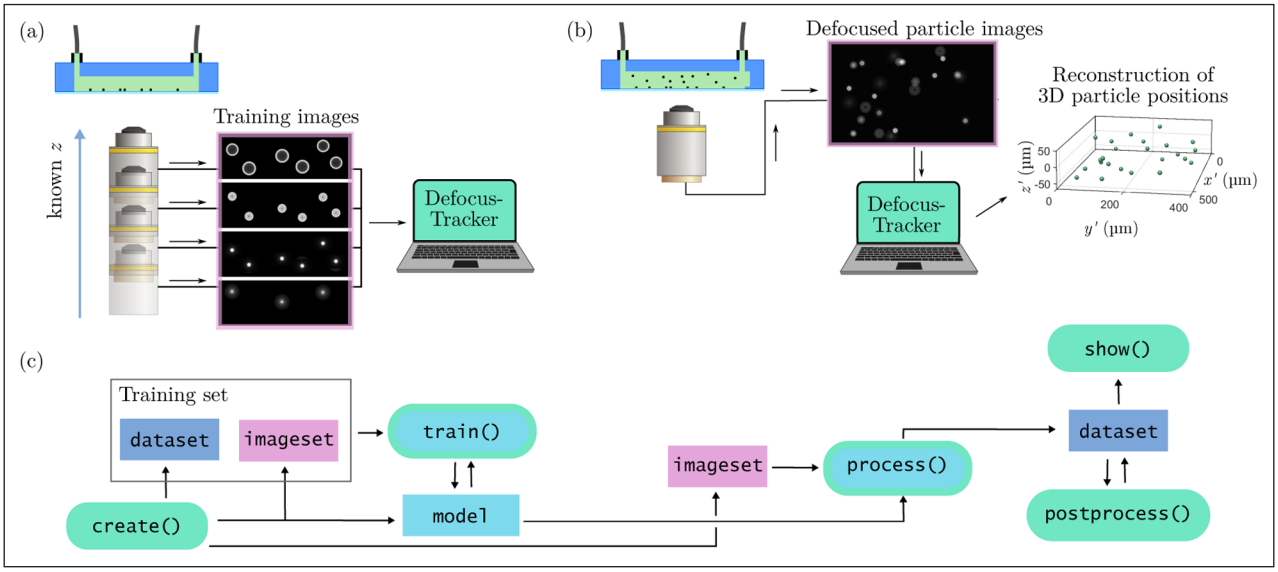

Figure 1 *DefocusTracker* working principle and general architecture. **(a)** A set of training images with known 3D particle positions are used to **(b)** determine the unknown 3D positions of particles through the comparison of their defocused particle images. **(c)** General architecture and workflow of the *DefocusTracker* toolbox. The toolbox uses three types of data structures (rectangles) and five main functions (round shapes, green). The toolbox is modular, allowing for addition and use of different models for the image processing and particle tracking.

Data structures:

`imageset` Link and description to a set of images, i.e. image paths, number of images, and image type.

`dataset` Particle data, i.e. 3D spatial positions and displacements, trajectories, connection to image frames, and detection accuracy estimation.

`model` Data and settings required to process the images, i.e. method-specific parameters as well as training and processing settings.

Functions:

`create()` Creates a data structure.

`show()` Opens GUIs to inspect data structures.

`train()` Trains a `model` on a specific training set (`imageset+dataset`).

`process()` Processes an `imageset` using a given `model`.

`postprocess()` Manipulates a `dataset`, e.g. merge datasets, apply scaling, remove outliers, perform particle tracking, filter trajectories, and estimate uncertainties.

Workflow:

As seen in ***Figure 1(c)***, a typical workflow starts with the creation of a `model` using the `create()` function. Each `model` refers to a specific method (e.g. Method 1) and it is at first created with some default values. The `model` is trained using the `train()` function by feeding a training set as input. A training set consists of a `dataset` and an `imageset` (made with `create()`), corresponding to a set of images containing one or more particles of known 3D position. As illustrated in ***Figure 1(a)***, such a training set can be obtained experimentally by taking subsequent images of particles displaced at known positions, e.g. by observing particles sedimented on a microchannel bottom, while taking images at known objective distances using a focusing stage. If only a subset of the calibration particles are used for a training, the remaining particles can be used as validation to make a pre-measurement uncertainty estimation.

With a trained `model` as input, the `process()` function can take one or more measurement images in an `imageset` and output a `dataset` containing the measured 3D particle positions. The `dataset` can be further manipulated via the `postprocess()` function for purposes such as outlier removal or trajectory smoothing. Throughout the entire workflow, the `show()` function can be used to visualize and inspect the data structures.

An example of typical *DefocusTracker* input (`imageset`) and output (`dataset`) is given in the online repository in the Work-Through Example 0 (WTE0).

Modularity:

The *DefocusTracker* toolbox is modular in the sense that a `model` can be created based on different methods for the image recognition. If a new method is added to the toolbox, the data structures, primary functions, and workflow will remain the same. The current implementation contains two methods for the creation of a `model`, namely boundary_threshold_2d (or Method 0) and normalized_crosscorrelation_3d (or Method 1).

Method 0: boundary_threshold_2d

Method 0 is used for 2D particle tracking based on setting a boundary intensity threshold to detect particles. This method provides a quick way to perform 2D tracking, which e.g. can be useful in the creation of training dataset using images with in-plane particle motion.

Method 1: normalized_crosscorrelation_3d

Method 1 performs the full defocusing-based 3D particle tracking, for a full description of Method 1, we refer to [15]. Briefly, Method 1 is based on training images of a single particle (the so-called calibration stack) and uses the normalized cross-correlation function for image recognition. The normalized cross-correlation is used to rate the similarity between a target particle image and the calibration stack images, using its maximum peak value as the similarity coefficient, referred to as C_m. The values of C_m can range from 0 to 1, with 1 corresponding to a perfect match between the target image and a calibration image.

Method X: Integrate a new method in *DefocusTracker*

In order to integrate a new method into the *DefocusTracker* toolbox, one must use the *DefocusTracker* infrastructure. This means that one must build a create-function to generate a template of the model structure as well as a process-function that as input takes the model structure and a *DefocusTracker* `imageset` and as an output gives a *DefocusTracker* `dataset`. For instance:

```
mymodel = mymethod_create()
mydataset = mymethod_process(mymodel,
myimagset, frame_index)
```

Here, all the parameters in the model structure must be organized in the following three mandatory subfields:

`mymodel.parameter` (Internal non-editable parameters)

`mymodel.training` (User-editable parameter settings for the training)

`mymodel.processing` (User-editable parameter settings for the processing)

MATLAB IMPLEMENTATION

DefocusTracker is implemented in MATLAB and additionally requires the image processing, curve fitting, and statistics toolboxes. The implementation is script-based with certain features using GUI-based

pop-up windows for visualization. The data structures are so-called MATLAB structs, namely structure arrays where data is grouped using containers called fields. The data in a field is accessed using dot notation of the form `structName.fieldName`, e.g. the path of an `imageset` is called with `imageset.path`. The primary toolbox functions follow the form of standard MATLAB functions and are named as `dtracker_functionName()`, e.g `dtracker_create()`. A full overview of the data structures and functions are shown in ***Figure 2***.

The MATLAB toolbox contains three work-through examples (WTE0, WTE1, and WTE2) that serve as tutorials to get new users quickly started, but also as test scripts in case new functionalities or methods are added. The scripts of each WTE are included in the *DefocusTracker* package, while the related datasets can be acquired via *https://defocustracking.com/datasets/*. WTE1 is based on synthetic images and gives a first introduction to the basic building blocks of the toolbox. The use of synthetic images allow for an exact estimation of the uncertainty using the postprocessing method `'compare_true_values'`. WTE2 is based on a state-of-the art microfluidic experiment, namely the 3D flow inside an evaporating droplet [17], and guides the user toward a more advanced use of *DefocusTracker*, including postprocessing and bias correction. We report in ***Figure 3*** a shortened version of the WTE2 script, including few screenshots of GUI panels obtained with the `dtracker_show()` function. For the full commented version we refer to the script `Work_through_ex2.m` as well as to *https://defocustracking.com/defocustracker*, where a published version of the code and output can be found. As the community grows, we expect that more WTEs will be added by the users and developers.

PYTHON IMPLEMENTATION

A Python implementation of *DefocusTracker* is planned and under development and it will follow the lines of the MATLAB implementation. The data structures will be implemented using Python dictionaries, whereas the functions will be part of the module `dtracker`. Following the above example, the path of an `imageset` will be called in Python with `imageset['path']`, whereas the `create()` function will be called as `dtracker.create()`.

The Python implementation is available on the Gitlab repository: *https://gitlab.com/defocustracking/defocustracker-python*. Updates and release information can be followed on *https://defocustracking.com*.

QUALITY CONTROL

The MATLAB toolbox has been tested functionally on Windows 10 with MATLAB releases R2018b, 2020a, and 2020b, while the toolbox performance has been tested and investigated extensively in three recent publications [2, 5, 15]:

Main data structures		
type	class	fields
imageset	struct	path, images, n_frames, im_type
dataset	table	fr, X, Y, Z, DX, DY, DZ, id, cm
model	struct	method, parameter training, processing, tracking, scaling

function dtracker_show(var, varargin)	
var	varargin
imageset	---
	dataset
dataset	---
	'plot_3d'
	'plot_3d_tracks'
model	---

function mystruct = dtracker_create(what_to_create, varargin)		
mystruct	what_to_create	varargin
imageset	'imageset'	---
		folder_name
dataset	'dataset'	---
		number of zeros in arrays
model	'model'	---
		'mymethod' (default 'normalized_crosscorrelation_3d')
tracking	'tracking'	---
		'mytracking' (default: 'nearest_neighbor')
scaling	'scaling'	---
		'myscaling' (default: 'linear')

function model = dtracker_train(model, imageset, dataset)

function dataset = dtracker_process(model, imageset)
dataset = dtracker_process(model, imageset, frame_index)

function varargout = dtracker_postprocess(what_to_do, dataset, varargin)			
varargout	what_to_do	dataset	Varargin
dataset	tracking	dataset	---
		dataset	frame_index
errors, errors_delta	'compare_true_values'	dataset	dataset_true
dataset	'merge_id'	dataset	dataset2
dataset	'min_cm'	dataset	min_cm
dataset	'min_n_id'	dataset	min_n_id
dataset	'scale'	dataset	---
			scaling
dataset	'unscale'	dataset	---
dataset	'update_id'	dataset	---

Figure 2 Overview of data structures and functions in the *DefocusTracker* MATLAB implementation, following the general toolbox architecture shown in Figure 1.

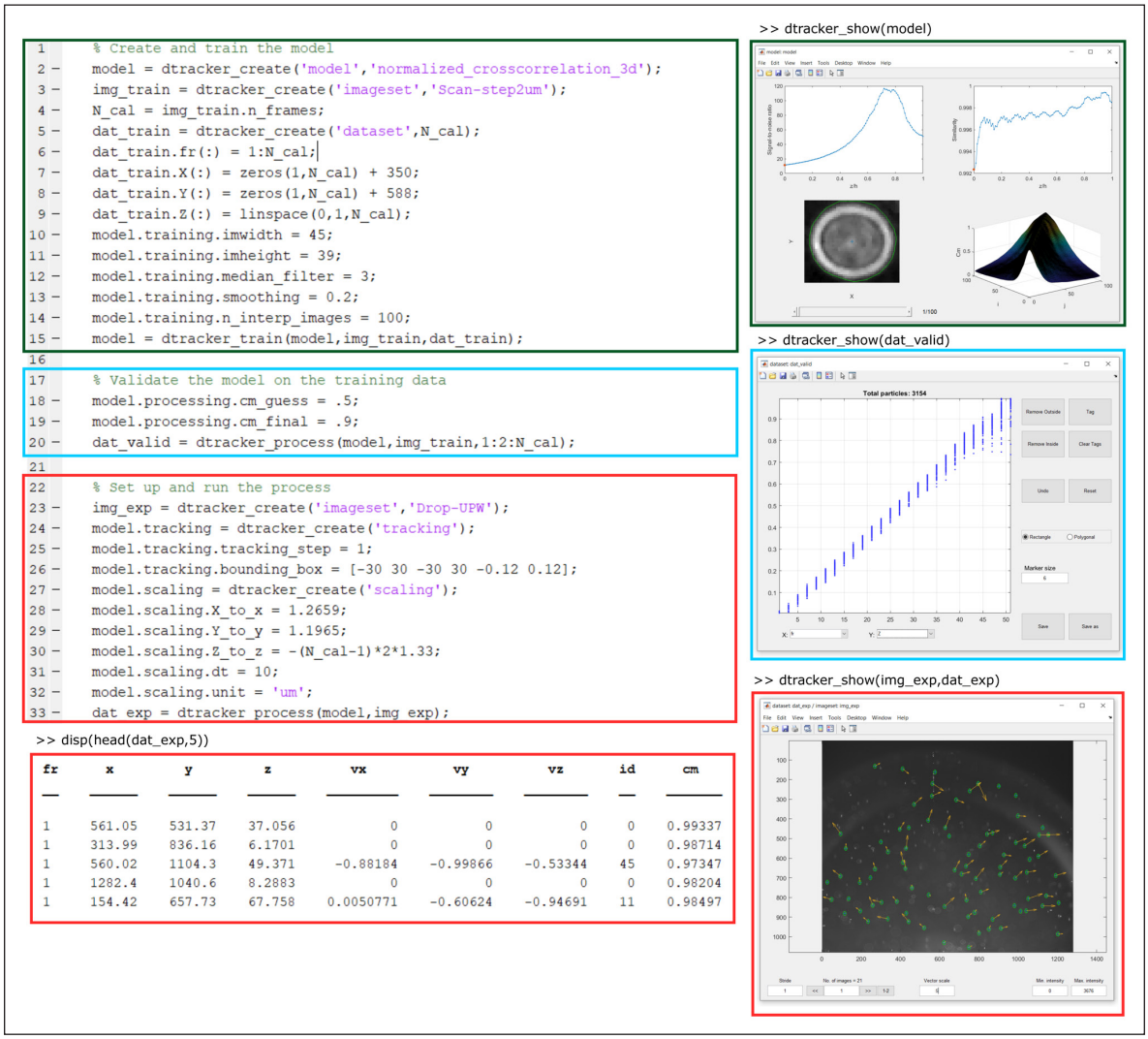

Figure 3 Example workflow of the toolbox MATLAB implementation. The example workflow is based on part of the provided Work-Through Example 2 (WTE2) that takes the user through the processing and analysis of particle trajectories inside an evaporating droplet [17]. The programming lines illustrate the code used to create and train a model (green frame), to validate the model on the training data (blue frame), and to process the measurement images (red frame). The frames of corresponding colors illustrate the implemented pop-up GUIs and tables used to visualize and inspect the data structures.

- In Barnkob and Rossi [5], guidelines for assessing the uncertainty of GDPT analyses were given. Synthetic images were used to test the toolbox in terms of measurement uncertainty and relative number of measured particles as a function of image signal-to-noise ratio, particle image concentration, and variations in image intensity.
- In Rossi and Barnkob [15], different toolbox settings were tested on synthetic and experimental images to outline the measurement uncertainties, detection rates, and processing times. The results were benchmarked against the GDPTlab software [9], which has been extensively-tested and used in high-impact research publications, see more in the Section Reuse potential.
- In Barnkob et al. [2], the toolbox was validated and compared against other defocus-tracking approaches and algorithms based on model functions and machine learning for image recognition. The different approaches were applied to synthetic and experimental images of different degrees of astigmatism, noise levels, and particle image overlapping. ***Figure 4*** summarizes the results when applying *DefocusTracker* to the synthetic image sets. ***Figure 4(a)*** shows example field of views of the analyzed synthetic images, while ***Figure 4(b)*** shows the synthetic particle images at different depth coordinates z over the measurement depth h. ***Figure 4(c)*** shows the resulting coordinate uncertainties σ and recall ϕ as a function of different degrees of astigmatism, noise levels, and particle image concentration N_s (the higher, the more particle image overlapping). Depending on the particle image concentration and the achieved recall for the given C_m-value, the depth coordinate uncertainties σ_z vary around 1–2% of the total measurement depth h, while the in-plane coordinate uncertainties σ_x, σ_y vary around 0.2–0.4 pixel.

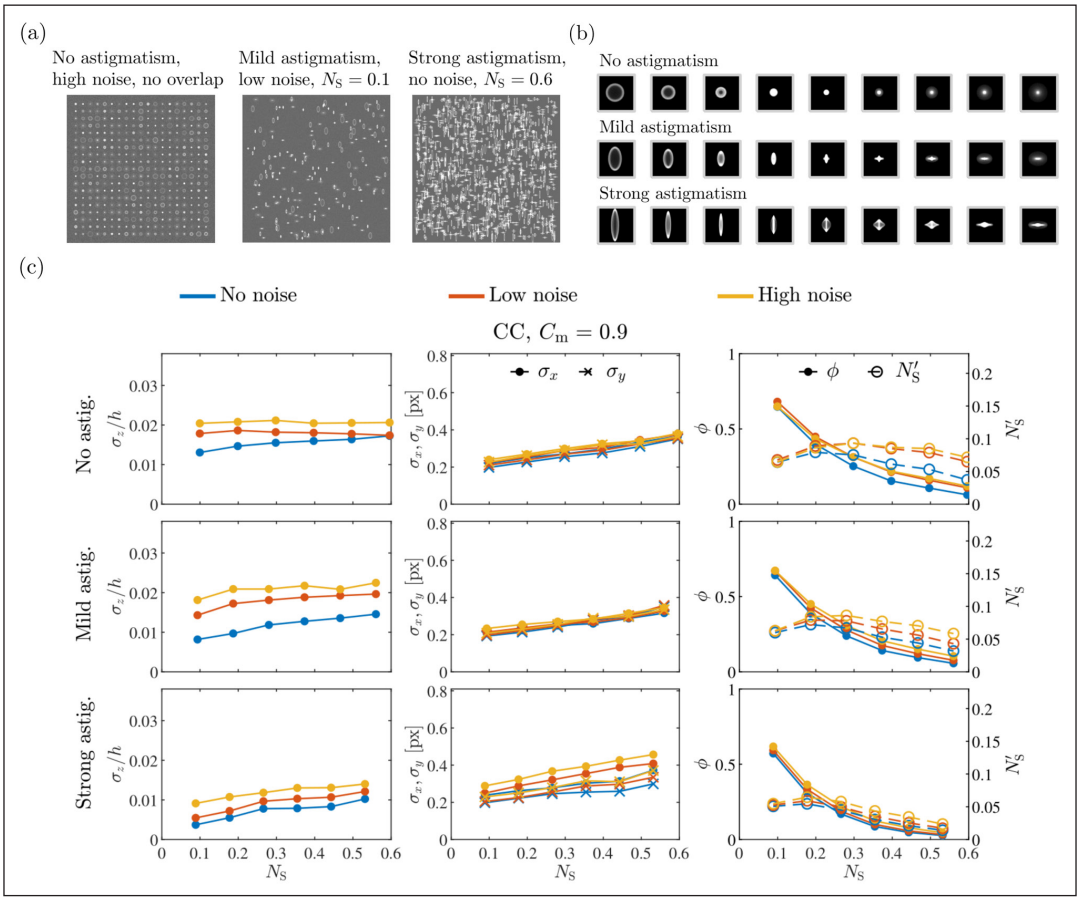

Figure 4 Example validation of the *DefocusTracker* MATLAB implementation as presented in Barnkob et al. [2], where different defocus-tracking approaches and algorithms were compared when applied to synthetic image sets of different degrees of astigmatism, noise levels, and particle image overlapping. **(a)** Example field of views of the analyzed synthetic images. **(b)** Example of the synthetic particle images at different z over the measurement depth h. **(c)** *DefocusTracker* results showing the coordinate uncertainties σ and recall ϕ.

(2) AVAILABILITY

OPERATING SYSTEM

Windows, UNIX/Linux, Macintosh (and any operating system supporting MATLAB).

PROGRAMMING LANGUAGE

MATLAB 9.4.0 (R2018a), upward compatible.

ADDITIONAL SYSTEM REQUIREMENTS

N/A

DEPENDENCIES

The MATLAB implementation requires the additional MATLAB toolboxes: 'curve_fitting_toolbox', 'image_toolbox', 'statistics_toolbox'

LIST OF CONTRIBUTORS

N/A

SOFTWARE LOCATION

Archive

Name: Gitlab

Persistent identifier: *https://gitlab.com/defocustracking/defocustracker-matlab/-/releases/v2.0.0*

Licence: MIT

Publisher: Massimiliano Rossi and Rune Barnkob

Version published: 2.0.0

Date published: 18/06/2021

Code repository

Name: Gitlab

Persistent identifier: *https://gitlab.com/defocustracking/defocustracker-matlab*

Licence: MIT

Date published: 18/06/2021

LANGUAGE

English

(3) REUSE POTENTIAL

The analysis of particle positions, velocities, and trajectories is an integral part of many research disciplines. This includes analyses in 2D as well as in 3D, and with the rapid growth in fields relying on microscopy, such single-camera methods can provide unique and important information. One example is within the field of microfluidics where

high control of fluid flow and externally-applied forces is becoming an important tool in biomedical research and applications. Here, 3D detection and tracking of particles and cells can provide the necessary information needed for optimization, standardization, and real-time inspection and control [18, 19].

GDPT has shown to be an excellent candidate for a wide-spread technique as is a simple and universal defocusing-based method and requires no special optics and can be used in standard microscope setups. Here, the development of free, accessible, user-friendly, and accurate tools can greatly enhance the practicability and availability of the method. One example is the MATLAB implementation GDPTlab (also by the authors), which has been distributed to researchers since 2015 and has proven its value in a number of research projects including work in journals such as Proceedings of the National Academy of Sciences, Physical Review Letters, and Scientific Reports [1, 4, 6–8, 10–14, 16, 21–23]. Note that most of this research were done in laboratories with no previous experience of 3D single-camera particle tracking prior to the use of GDPTlab. GDPTlab has thus shown the huge potential such methods hold, if free and user-friendly implementations are available. Though GDPTlab has enabled many researchers to get started using GDPT, it has unfortunately not been fully accessible as an open-source project, limiting its further development and adaptation by the community. *DefocusTracker* fills this need as it is fully open-source. The toolbox contains a readme-file and work-through examples for users to get quickly started.

MODIFICATION AND SUPPORT

DefocusTracker is set up in a versatile and modular fashion allowing for easy expansions and improvements, such as extensions of custom functionalities and features, e.g. using MATLAB's GUI editor and pre-built functions. In the Python implementation, such expansions could involve the use of popular libraries for data analysis and machine learning, such as SciKit, Keras, and TensorFlow. *DefocusTracker* is supported by *https://defocustracking.com/* which is an online platform created to assist the development and support of *DefocusTracker* as well as to facilitate the research community with a place for sharing of data and experiences related to single-camera 3D particle tracking. The platform contains several forums, e.g. for new developers to request access and for users to ask the community for support.

DATA REPOSITORY

All material, without exception, is available via the permanent repository: *https://defocustracking.com/.*

ACKNOWLEDGEMENTS

We would like to thank all GDPTlab and *DefocusTracker* users for their input and comments. In particular, we would like to thank our colleague Prof. Alvaro Marin for his invaluable comments and support from the very beginning of the GDPT project.

FUNDING STATEMENT

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement no. 713683 (COFUNDfellows-DTU).

COMPETING INTERESTS

The authors have no competing interests to declare.

AUTHOR AFFILIATIONS

Rune Barnkob *orcid.org/0000-0003-1282-4990*
Heinz-Nixdorf-Chair of Biomedical Electronics, Department of Electrical and Computer Engineering, Technical University of Munich, Center for Translational Cancer Research (TranslaTUM), Munich, Germany

Massimiliano Rossi *orcid.org/0000-0002-6579-7132*
Department of Physics, Technical University of Denmark, DTU Physics Building 309, DK-2800 Kongens Lyngby, Denmark

REFERENCES

1. **Ashaju AA, Wood JA, Lammertink RG.** Electrocatalytic reaction-driven flow. *Physical Review Fluids*. 2021; 6(4): 044004. DOI: *https://doi.org/10.1103/PhysRevFluids.6.044004*
2. **Barnkob R, Cierpka C, Chen M, Sachs S, Mäder P, Rossi M.** Defocus particle tracking: A comparison of methods based on model functions, cross-correlation, and neural networks. *Measurement Science and Technology*. 2021; 32(9): 094011. DOI: *https://doi.org/10.1088/1361-6501/abfef6*
3. **Barnkob R, Kähler CJ, Rossi M.** General defocusing particle tracking. *Lab on a Chip*. 2015; 15(17): 3556–3560. DOI: *https://doi.org/10.1039/C5LC00562K*
4. **Barnkob R, Nama N, Ren L, Huang TJ, Costanzo F, Kähler CJ.** Acoustically driven fluid and particle motion in confined and leaky systems. *Physical Review Applied*. 2018; 9(1): 014027. DOI: *https://doi.org/10.1103/PhysRevApplied.9.014027*
5. **Barnkob R, Rossi M.** General defocusing particle tracking: fundamentals and uncertainty assessment. *Experiments in Fluids*. 2020; 61: 1–14. DOI: *https://doi.org/10.1007/s00348-020-2937-5*

6. **Bodé WN, Jiang L, Laurell T, Bruus H.** Microparticle acoustophoresis in aluminum-based acoustofluidic devices with pdms covers. *Micromachines*. 2020; 11(3): 292. DOI: *https://doi.org/10.3390/mi11030292*
7. **Gelin P, Maes D, De Malsche W.** Reducing Taylor-Aris dispersion by exploiting lateral convection associated with acoustic streaming. *Chemical Engineering Journal*. 2021; 417: 128031. DOI: *https://doi.org/10.1016/j.cej.2020.128031*
8. **Giuliani N, Rossi M, Noselli G, DeSimone A.** How euglena gracilis swims: flow field reconstruction and analysis. *Physical Review E*. 2021; 103(2): 023102. DOI: *https://doi.org/10.1103/PhysRevE.103.023102*
9. *GDTPlab|how to get it, Institut für Strömungs-mechanik und Aerodynamik, Univeristät der Bundeswher, Werner Heisenberg Weg 39, 85577 Neubiberg, Germany* [n.d.], *https://www.unibw.de/lrt7/gdpt-1/gdptlab-how_to_get_it*. Accessed: 3 June 2020.
10. **Karlsen JT, Qiu W, Augustsson P, Bruus H.** Acoustic streaming and its suppression in inhomogeneous fluids. *Physical review letters*. 2018; 120(5): 054501. DOI: *https://doi.org/10.1103/PhysRevLett.120.054501*
11. **Liu J, Reisbeck M, Hayden O.** Investigation of mechanical and magnetophoretic focusing for magnetic flow cytometry. *Current Directions in Biomedical Engineering*. 2019; 5(1): 353–355. DOI: *https://doi.org/10.1515/cdbme-2019-0089*
12. **Qiu W, Bruus H, Augustsson P.** Particle-size-dependent acoustophoretic motion and depletion of micro-and nano-particles at long timescales. *Physical Review E*. 2020; 102(1): 013108. DOI: *https://doi.org/10.1103/PhysRevE.102.013108*
13. **Qiu W, Karlsen JT, Bruus H, Augustsson P.** Experimental characterization of acoustic streaming in gradients of density and compressibility. *Physical Review Applied*. 2019; 11(2): 024018. DOI: *https://doi.org/10.1103/PhysRevApplied.11.024018*
14. **Rahman MRU, Kwak TJ, Woehl JC, Chang W-J.** Quantitative analysis of the three-dimensional trap stiffness of a dielectrophoretic corral trap. *Electrophoresis*. 2021; 42(5): 644–655. DOI: *https://doi.org/10.1002/elps.202000222*
15. **Rossi M, Barnkob R.** A fast and robust algorithm for general defocusing particle tracking. *Measurement Science and Technology*. 2020; 32(1): 014001. DOI: *https://doi.org/10.1088/1361-6501/abad71*
16. **Rossi M, Cicconofri G, Beran A, Noselli G, DeSimone A.** Kinematics of agellar swimming in euglena gracilis: Helical trajectories and agellar shapes. *Proceedings of the National Academy of Sciences*. 2017; 114(50): 13085–13090. DOI: *https://doi.org/10.1073/pnas.1708064114*
17. **Rossi M, Marin A, Kähler CJ.** Interfacial flows in sessile evaporating droplets of mineral water. *Physical Review E*. 2019; 100(3): 033103. DOI: *https://doi.org/10.1103/PhysRevE.100.033103*
18. **Sapudom J, Waschke J, Franke K, Hlawitschka M, Pompe T.** Quantitative label-free single cell tracking in 3d biomimetic matrices. *Scientific reports*. 2017; 7(1): 1–9. DOI: *https://doi.org/10.1038/s41598-017-14458-x*
19. **Sitters G, Kamsma D, Thalhammer G, Ritsch-Marte M, Peterman EJ, Wuite GJ.** Acoustic force spectroscopy. *Nature methods*. 2015; 12(1): 47–50. DOI: *https://doi.org/10.1038/nmeth.3183*
20. **Taute K, Gude S, Tans S, Shimizu T.** High-throughput 3D tracking of bacteria on a standard phase contrast microscope. *Nature communications*. 2015; 6: 8776. DOI: *https://doi.org/10.1038/ncomms9776*
21. **Van Assche D, Reithuber E, Qiu W, Laurell T, Henriques-Normark B, Mellroth P, Ohlsson P, Augustsson P.** Gradient acoustic focusing of sub-micron particles for separation of bacteria from blood lysate. *Scientific reports*. 2020; 10(1): 1–13. DOI: *https://doi.org/10.1038/s41598-020-60338-2*
22. **Volk A, Kähler CJ.** Size control of sessile microbubbles for reproducibly driven acoustic streaming. *Physical Review Applied*. 2018; 9(5): 054015. DOI: *https://doi.org/10.1103/PhysRevApplied.9.054015*
23. **Westerbeek EY, Bomer J, Olthuis W, Eijkel JC, De Malsche W.** Reduction of taylor-aris dispersion by lateral mixing for chromatographic applications. *Lab on a Chip*; 2020. DOI: *https://doi.org/10.1039/D0LC00773K*

TO CITE THIS ARTICLE:
Barnkob R, Rossi M 2021 *DefocusTracker*: A Modular Toolbox for Defocusing-based, Single-Camera, 3D Particle Tracking. *Journal of Open Research Software*, 9: 22. DOI: *https://doi.org/10.5334/jors.351*

Submitted: 01 October 2020 Accepted: 07 July 2021 Published: 23 July 2021