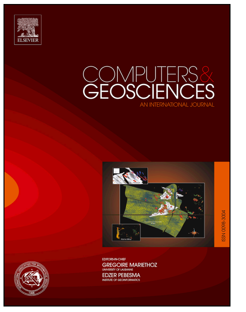
TracTrac: A fast multi-object tracking algorithm for motion estimation

Joris Heyman

Accepted Manuscript



TracTrac: A fast multi-object tracking algorithm for motion estimation

Joris Heyman

PII: S0098-3004(18)31066-5

DOI: https://doi.org/10.1016/j.cageo.2019.03.007

Reference: CAGEO 4254

To appear in: *Computers and Geosciences*

Received Date: 7 November 2018

Revised Date: 13 February 2019

Accepted Date: 22 March 2019

Please cite this article as: Heyman, J., TracTrac: A fast multi-object tracking algorithm for motion estimation, *Computers and Geosciences* (2019), doi: https://doi.org/10.1016/j.cageo.2019.03.007.

TracTrac: a fast multi-object tracking algorithm for motion estimation

Joris Heyman
Univ Rennes, CNRS
Géosciences Rennes, UMR 6118
35000 Rennes, France.
joris.heyman@univ-rennes1.fr

March 28, 2019

Abstract

TracTrac is an open-source Matlab/Python implementation of a robust and efficient object tracking algorithm capable of simultaneously tracking several thousands of objects in very short time. Originally developed as an alternative to particle image velocimetry algorithms for estimating fluid flow velocities, its versatility and robustness makes it relevant to many other dynamic sceneries encountered in geophysics such as granular flows and drone videography. In this article, the structure of the algorithmis detailed and its capacity to resolve strongly variable and intermittent object motions is tested against three examples of geophysical interest.

Keywords: Videography, Motion Estimation, Feature Tracking

1 Introduction

Owing to the popularization of digital cameras in the last 20 years, videography techniques are increasingly used in the lab and in the field to measure velocities and trajectories associated to a moving scenery. As earth processes are mostly dynamic, imaging today appears as an affordable way to get spatio-temporal quantification of these motions. Glacier motion, river flow, sediment transport, rock avalanches, wind boundary layers, are some example of geophysical processes, whose understanding rely deeply on how accurately their kinematics can be measured both in time and in space. Yesterday restricted to laboratory studies with important experimental apparatus (lasers, high speed cameras, computing clusters), flow imaging is now expanding to in-situ monitoring of geophysical processes, notably thanks to the new perspectives offered by drone videography [1]. In parallel, significant efforts have been made in the computer vision community to improve and invent new image processing algorithms treating efficiently these image sequences. Applications for video surveillance (such as face recognition) and autonomous vehicles are among the most spectacular achievements of these algorithms, running in real time [3, 14]. Curiously, few of these new methods were transferred into user-friendy, flexible and open-source applications available for earth science researcher in their daily work [5]. Processing images often costs much of the scientific effort instead of being a powerful and direct mean to better understand natural processes. The present article introduces TracTrac (see Computer Code Availability section), an open-source Matlab/Python implementation of an original and efficient object tracking algorithm capable of simultaneously tracking several thousands of objects in very short computation time and very basic user knowledge. Its conception makes it equally good for dealing with densely seeded fluid flows (typically treated with Particle Image Velocimetry methods, PIV), granular flow, birds motion or any natural moving scene. The article is structured as follows. After briefly reviewing the computational methods that have been proposed for motion estimation in the past, TracTrac originality and the algorithm structure is detailed. The accuracy and robustness of the algorithm is then tested against a synthetic images occupied by artificial moving objects. Finally, examples of TracTrac flow estimates are presented in real earth science applications (turbulence, granular avalanche and bird flock).

2 Advances in motion estimation techniques

Literature about motion estimation from image sequence is vast and spreads over several scientific disciplines, rendering difficult an exhaustive review. At least two large families of methods have emerged: (i) the methods based on interrogation windows, usually called Particle Image Velocimetry (PIV) and (ii) the methods based on object detection and tracking, typically called Particle Tracking Velocimetry (PTV). Although being not very informative, the choice of these acronyms refers to their initial use in films of flows seeded by tracer particles to establish a map of local velocities. The spectrum of applications of PIV and PTV methods is however much broader, extending to complex moving sceneries. The general idea behind PIV is to quantify motion by cross-correlation of interrogation windows [25]. Dividing the image into smaller boxes (typically of 8 or 16 pixels width), the local motion is obtained by searching the box displacement that maximizes the cross-correlation product of box pixel light or color intensities between two consecutive frames. In contrast, PTV consists in detecting the presence of special features in an image and tracking them through consecutive frames. Special features, also called objects, can be particles (blobs of bright or dark intensities), but also more complex shapes (corners, ball, faces, ...). Where PIV outputs a velocity vector for each interrogation box, PTV provides the trajectory of an object (its position in successive frames) which can be then mapped into a grid to get a dense velocity field [22]. In other words, PTV takes the Lagrangian point of view of motion where PIV is essentially an Eulerian vision. Both techniques have advantages and disadvantages as shown in the following. While originally preferred for its relative simplicity and robustness (a few free parameters involved), PIV inevitably introduces some filtering at fluctuation scales smaller than the box size, which preclude the correct estimation of steep velocity gradients. Partial recovering of interrogation boxes help at increasing velocity map resolution but do not solve the filtering effect. Uncertainties are thus particularly high for flows near walls and turbulent flows in general for which the Kolmogorov scale can be small. In contrast, PTV is only limited by the scale of the tracked features (particles, gradients) as well as their local density so that instantaneous velocity maps are less prone to the box filter effect [10, 11]. Both PIV and PTV dynamic ranges strongly rely on the accurate detection of peaks in the frames (e.g., the location of the feature to track for PTV and maximum cross-correlation product for PIV). Methods have been proposed to reach sub-pixel accuracy in peak location, allowing for the measurement of displacements smaller than one pixel per frame. However, local saturation of images (values equal to 0 or 1) and small particle image size may produce peak locking and biased velocity measurements [6, 15, 16, 19, 20, 23]. To minimize these effects, attention has to be taken to the effective dynamic range reached (high contrasts) and the magnification factor of the lens. PIV methods typically overtake PTV methods if particle displacement becomes large compared to the mean inter particle image distance. A so-called particle spacing displacement ratio $p = \sqrt{S/N}/(v\Delta t)$ (with $v\Delta t$ the particle image displacement, N the number of particles and S the image surface) has been proposed by [13] to describe this effect. To avoid ambiguities in the reconstruction of trajectories, PTV algorithms generally requires high frame rates or low particle image densities, that is $p \geq 1$. Thus, PIV algorithms have often been preferred to probe densely seeded turbulent flows where p can be small compared to 1. Combinations of the two methods have been proposed to gain robustness in the case of large displacements (or large particle image density) to limit the low-pass filtering effect of PIV [7, 21]. Another disadvantage of PIV methods concerns complex sceneries made of moving and non-moving layers (e.g. a flowing river on a fixed bed, a flock of flying birds through trees). The cross-correlation procedure do not differentiate between layers so that the resulting velocity is an average of the fixed and moving elements. In addition, incoherent motion such as Brownian motion or multiple wave celerities cannot be handled by most PIV methods: at the scale of the interrogation window, the flow is supposed to be continuous and unidirectional. In contrast, sharp interfaces between moving and static regions, as well as non-coherent motions can in principle be rendered by PTV methods [5]. There is today a net enthusiasm for PTV algorithms due to their broader application range and their higher resolution [5, 9]. They are also directly transferable to stereoscopic camera setup where the position and trajectory of objects can be estimated in the three space dimensions [12, 13, 18]. However, most existing PTV algorithms still suffer from the aforementioned drawbacks: (i) they are limited to large particle spacing displacement ratio ($p > 2$ in [13] and in [5], while p>0.33 in a recent study by [7]) and (ii) they are not computationally efficient when many features have to be tracked (a maximum of 4000 particles per time frame are considered in [7] and 1000 in [5]).

3 TracTrac

The first innovation brought by TracTrac compared to traditional PTV methods is its efficiency. Indeed, TracTrac uses k-dimensional trees to search and compute statistics around neighbouring objects [4], allowing very high analysis frame rate even at large particle image number. The second key feature lie in an original association process of objects between frames, that significantly decreases the number of erroneous trajectory reconstructions. This process is based upon a sequence of 3 frames (instead of 2 for classical pair association) and a conservative rule rejecting any association ambiguities [21]. The third advantage of TracTrac is its capacity to deal with high feature densities at relatively low acquisition frequencies (p down to 0.25). This is achieved owing to a motion predictor step based on a local spatio-temporal average of the neighbouring object velocities. Differences between the motion prediction model and the observed displacement are systematically monitored, allowing filtering outliers based on local and adaptive statistics of the motion variability. This adaptive filter enables both the quantification of strongly incoherent motions (of the Brownian motion type [5]) and coherent displacement (governed by a spatio-temporal continuous deterministic velocity field) in the same image.

3.1 Details of the algorithm

TracTrac rests on three main modules: object detection, motion estimation, error monitoring.

3.1.1 Object detection

The first module regroups all the computing steps from the raw frame I_t to the detection of the position of moving objects $\boldsymbol{x}_t$. Most of these preprocessing steps are optional, and may be turned off by the user. The procedure is the following. First a median box filter can be applied to remove possible noise on I_t. The default size of this filter was set to 3×3 pixels. Second, the image is divided between a "background" image B_t made of quasi-static regions, and a "foreground image" F_t, formed by the moving regions. The latter is computed as $F_t = I_t - B_t$. This operation allows focusing on the moving part of a scene, while ignoring the static regions. B_t has to be recomputed at each frame. The method chosen here borrows from the so-called "median" background subtraction method, where the background image is taken to be a temporal moving average of pixel values:

$$B_t = \beta \, \mathrm{sign}(I_t - B_{t-1}) + B_{t-1}, \tag{1}$$

where $\beta < 1$ is the background adaptation speed. A large value of β gives backgrounds that are rapidly adapted to the changes in scene luminosity. On opposite, a small value of β provides background images that are insensible to rapid luminosity changes. The default value is set to $\beta = 0.001$. The recurrence relation (1) requires to provide an initial guess for B_0, which is computed from an average of the first $1/(2\beta)$ frames

$$B_0 = \frac{2}{\beta} \sum_{t=0}^{\beta/2-1} I_t. \tag{2}$$

It is worth noting that PTV methods are able to resolve sharp velocity gradients as well as out-of-plane velocity gradients so that background subtraction may not be always necessary. Objects are then identified in the foreground image by a so-called "blob detection" method. TracTrac integrates two state-of-the-art detectors, namely Difference of Gaussians (DoG) and Laplace of Gaussian (LoG), both depending on a single scale parameter δ. The DoG convolves the image with a filter constructed from the subtraction of two Gaussian of bandwidth 0.8δ and 1.2δ. It acts as a band-pass filter selecting blobs in the 20% scale range around $\delta\sqrt{2}$. The LoG approach first convolves the image with a Gaussian filter of bandwidth δ, then applying the Laplacian operator on the convoluted image. Both approaches yield a filtered image F'_t with a strong positive response in the presence of objects of scale δ. Positions of the object centroids $\{\boldsymbol{x}_t\}$ are obtained by searching for local maximum in F'_t. To minimize false detections, an intensity threshold ε is fixed under which maxima are ignored. In TracTrac, the default value of ε is fixed to half standard deviation above the mean luminosity of F'_t. Sub-pixel resolution of object position is achieved by fitting a quadratic or a Gaussian function to the pixel intensity values around the centroid position, and finding then the position of the maximum of this function. For instance, if a maximum is found in pixel i, j, the sub-pixel position of object will be

$$x = j + \frac{F'_{i,j+1} - F'_{i,j-1}}{2(F'_{i,j+1} - 2F'_{i,j} + F'_{i,j-1})}, \tag{3}$$

$$y = i + \frac{F'_{i+1,j} - F'_{i-1,j}}{2(F'_{i+1,j} - 2F'_{i,j} + F'_{i-1,j})}, \tag{4}$$

$$\tag{5}$$

for a quadratic function. The formula is the same for a Gaussian function, replacing F' by $\ln(F')$. The ensemble of points $\boldsymbol{x}_i(t)$, $i = 1, \ldots, N(t)$ made of the sub-pixel centroid positions are then tracked through time.

3.1.2 Motion estimation

As classical PTV algorithms [7, 13], motion estimation is achieved by associating detected objects between successive frames, typically by minimization of Euclidean distance. In TracTrac, at each time t, the set of $N(t)$ detected objects is organized into a 2-dimensional tree allowing for fast nearest neighbour search [4]. The nearest neighbours in successive frames are computed for both forward and backward time association:

- forward $\boldsymbol{x}_i(t) \rightarrow \boldsymbol{x}_j(t+1)$: for each $\boldsymbol{x}_i(t)$, find its closest neighbour in $\{\boldsymbol{x}(t+1)\}$,
- backward $\boldsymbol{x}_i(t+1) \rightarrow \boldsymbol{x}_j(t)$: for each $\boldsymbol{x}_i(t+1)$, find its closest neighbour in $\{\boldsymbol{x}(t)\}$.

Since objects usually appear and disappear through frame, both computations may give different results. In order to minimize false associations, only the unequivocal pairs are kept (i.e., the pairs that point to the same objects regardless of the time direction of association). In doing so, ambiguous associations are automatically disregarded. A fragment of trajectory ("tracklet") is defined if two consecutive and unequivocal associations are made for the same object. In other words, when a position triplet $x_i(t-1) \leftrightarrow x_j(t) \leftrightarrow x_k(t+1)$ is found without ambiguity, it is considered as a valid fragment of trajectory, to which is associated a new or existing ID (depending on whether the object has been already associated to a trajectory ID in the frame $t-1$). This 3-frame association technique reduces significantly the occurrence of bad associations. In addition, it enables the computation of second order object velocities (via central differences) as well as their accelerations:

$$\hat{\boldsymbol{v}}(t-1) = \frac{\hat{\boldsymbol{x}}(t) - \hat{\boldsymbol{x}}(t-2)}{2\Delta t}, \tag{6}$$

$$\hat{\boldsymbol{a}}(t-1) = \frac{\hat{\boldsymbol{x}}(t) - \hat{\boldsymbol{x}}(t-2) + 2\hat{\boldsymbol{x}}(t-1)}{\Delta t^2} \tag{7}$$

This technique does not increase the computational cost significantly since nearest neighbour associations $(x_j(t) \leftrightarrow x_k(t+1))$ are saved for the following time step. In the following, the variables pertaining to objects that were associated into tracklets in the frame t are denoted by a hat (i.e., $\hat{\boldsymbol{x}}(t)$, $\hat{N}(t)$). The quality of this association step is often constrained by the maximum object displacement between consecutive frames, or equally the maximum object velocity divided by the frame rate of the camera. Indeed, erroneous associations spontaneously arise from aliasing effects when object displacement is comparable to the average distance separating objects (for instance, points on a line distant by 10 pixels that travel at 10 pixels per frame will appear having a null velocity). Motion recognition is relatively easy when $p^{-1} = v\Delta t\sqrt{N/S} \ll 1$ (or equally when $p \gg 1$ [13]). In TracTrac, this condition is relaxed by the use of a predictor step based on a motion model inherited from previous time step [7, 17]. In other words, TracTrac first predicts the position objects in the following frame and then use this prediction to perform the following association. At time t, the motion model is based on the pool of objects associated to tracklets at $t-1$, their velocities $\hat{\boldsymbol{v}}(t-1)$ and their motion predictors $\bar{\hat{\boldsymbol{v}}}(t-1)$ (where the bar symbol

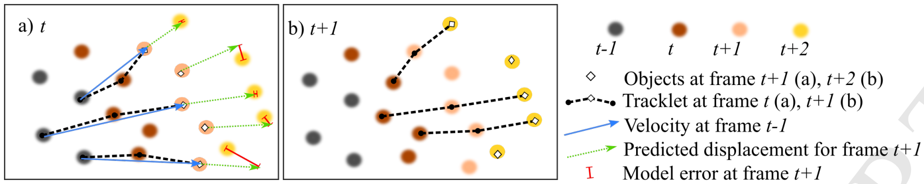


Figure 1: Prediction-Association process between frames t, $t+1$ and $t+2$

stands for quantities related to the motion model). This information is passed to all objects detected in the current frame t by a weighted average over their k-th nearest neighbours taken from the aforementioned pool:

$$\overline{\boldsymbol{v}}_i(t) = \frac{1}{\min(k,\hat{N})} \sum_{j=1}^{\min(k,\hat{N})} \alpha \hat{\boldsymbol{v}}_{i,j}(t-1) + (1-\alpha)\hat{\overline{\boldsymbol{v}}}_{i,j}(t-1). \tag{8}$$

The weight $\alpha \in [0,1]$ introduces a finite temporal relaxation of the predicted velocities. For $\alpha \to 1$, the motion model is only based on the immediate previous frame, while for $\alpha \to 0$, history of the velocities predicted in earlier frames are used to compute the motion model. Averaging over the k-th nearest neighbours has two advantages. First, it allows filtering the smallest spatial variations of velocities, that can be influenced by noise or erroneous tracklets associations. Second, it naturally adapts to the local density of objects, in contrast to fixed-size kernel smoothing methods: the larger the density, the smaller the filtering scale, and the finer the prediction. Once the motion model is computed, new object position is predicted assuming zero acceleration:

$$\overline{\boldsymbol{x}}_i(t) = \boldsymbol{x}_i(t-1) + \overline{\boldsymbol{v}}_i(t)\Delta t, \tag{9}$$

and the association process is performed by searching among the nearest neighbours between $\overline{\boldsymbol{x}}(t)$ and $\boldsymbol{x}(t+1)$ (Fig. 1). These new tracklets can either be saved and the following frame proceed, or used iteratively to refine the motion model and predict once again object displacement. The predictor step is thus implemented as an iterative sequence, using the temporary recovered tracklets as additional velocity vectors considered in the motion model. Convergence is generally obtained after a few iterations, the number of associated tracklet reaching a maximum. Once the desired number of iteration is reached, computation continues with the following frame.

3.1.3 Error monitoring and outliers filtering

The motion model used in TracTrac enables a continuous monitoring of the difference between predicted and actual displacements. This information is of particular value since it helps to eliminate outliers from the obtained associations based on statistical criterion. For each unequivocal associations, the log-error norm between the predicted and the true velocity vector is

$$\epsilon_i(t) = \log\left(\left\|\hat{\overline{\boldsymbol{v}}}_i(t-1) - \hat{\boldsymbol{v}}_i(t)\right\|\right). \tag{10}$$

which may be considered a real valued spatio-temporal random variable of approximately Gaussian shape (while the error norm would be log-normal since positive). Negative ϵ corresponds to high motion model accuracy. The probability distribution of ϵ depends on the spatial and temporal variability of the background flow to be measured as well as the quality of the zero acceleration approximation for the motion model. The local mean model error $\overline{\epsilon}_i$ around the object i is estimated by sampling log-errors over its k-nearest objects, in the same time as determining model velocities (8):

$$\overline{\epsilon}_i(t) = \frac{1}{\min(k,\hat{N})} \sum_{j=1}^{\min(k,\hat{N})} \alpha\hat{\epsilon}_{i,j}(t) + (1-\alpha)\hat{\overline{\epsilon}}_{i,j}(t). \qquad (11)$$

The standard deviation of the error around the mean is estimated on the whole computation window by

$$\overline{\sigma}_\epsilon(t) = \alpha\sqrt{\frac{1}{N(t)} \sum_{i=1}^{N(t)} \left(\hat{\epsilon}_i(t) - \frac{1}{N(t)} \sum_{j=1}^{N(t)} \hat{\epsilon}_j(t)\right)^2} + (1-\alpha)\overline{\sigma}_\epsilon(t-1), \qquad (12)$$

Outliers are then detected from tracklets which have $\epsilon_i(t) - \overline{\epsilon}_i(t) > n_\sigma\overline{\sigma}_\epsilon(t)$, with $n_\sigma \in \mathrm{R}$ a parameter chosen by the user. For instance, $n_\sigma = 1.96$ ensures that all associated tracklets remain in the 95% confidence interval provided by the model. In contrast, for $n_\sigma = -1.96$ only remains the 5% of tracklets that best fit the prediction model.

3.1.4 From tracklets to trajectories

To each new associated tracklet is given a trajectory ID number. If in the following frame, a tracklet is found with an object already having an ID, the latter is applied to the tracklet. This information handover allows reconstructing the whole object trajectories from elementary tracklets sharing the same ID. At each frame, the information about tracklets are saved by TracTrac in an array with columns: At the end of

Column Number	1	2	3	4	5	6	7	8	9	10	11
Variable	t	ID	$\hat{x}_i$	$\hat{y}_i$	$\hat{u}_i$	$\hat{v}_i$	$\hat{a}_{x,i}$	$\hat{a}_{y,i}$	$\hat{\overline{u}}_i$	$\hat{\overline{v}}_i$	$\hat{\overline{\epsilon}}_i$

Table 1: Correspondence between columns and variables in the TracTrac output ASCII file "`*_track.txt`"

the computation, this array can be saved either in ASCII or in binary format (mat-file in Matlab, and hdf5 in python). This file is automatically named according to the video file name with the suffix "`*_track.txt`".

3.2 User interface

The Matlab version of TracTrac includes a graphical interface (GUI) enabling rapid tracking results for non-expert users. In practice, it can be used to test and optimize the free parameters meanwhile observing in real time their effect on the quality of the tracking process. In contrast to the Matlab GUI, the Python version of TracTrac can be launched either as a Python script or as a Python function. This command line control allows treating iteratively several videos or integrating TracTrac directly into Python scripts (a list of video files can also be chosen in the Matlab GUI). Full compatibility is maintained between the two implementations owing to a common input parameter file whose structure is given as the supplementary material. Details about the Matlab GUI and the Python commands are also provided in this document.

4 Results and discussion

4.1 Synthetic flow

In order to test TracTrac performances, synthetic images were created, enabling a comparison of the algorithm predictions with known object trajectories. The flow was chosen in order to test the algorithm robustness for both strongly unsteady and non-uniform continuous flow field.

4.1.1 Flow description

The synthetic trajectories are initiated by N points randomly distributed in the image (x_0, y_0). At each frame, a synthetic image is build by applying a Gaussian kernel of fixed standard deviation on each object centroid. Uncorrelated noise is then added to the image pixels with an intensity depending on the signal-to-noise ratio (SNR) chosen (Fig. 2). An image is created at each frame, while advecting the objects according to the following two consecutive operations: a first one operating in radial coordinates ($r = \sqrt{x^2 + y^2}, \theta = \tan^{-1}(y/x)$):

$$r_{n+1} = r_n, \tag{13}$$

$$\theta_{n+1} = \theta_n + 4\delta\cos(n\pi/50)\exp(-0.5(r_n/80)^2) - 10\delta\cos(n\pi/25)\exp(-0.5(r_n/50)^2), \tag{14}$$

followed by a second step in cartesian coordinates ($x = r\cos\theta, y = r\sin\theta$):

$$x_{n+1} = x_n + 2\delta\sin(\pi n/100) * y_n + \xi_n, \tag{15}$$

$$y_{n+1} = y_n + \delta x_n + \xi_n, \tag{16}$$

where ξ_n is a white noise term whose intensity will be varied. In practice, the time step $\delta = 0.01$ is chosen to get displacement lengths in the range 0-20 pixels per frames. Cartesian and polar coordinate systems are centred on the image centre. Periodic boundary conditions are applied when a point leaves the image field. Snapshot of the flow velocity vectors were plotted on Fig. 3. Before the next frame, a finite number of points (0-50%) are associated new coordinates in order to mimic in and out of plane motion.

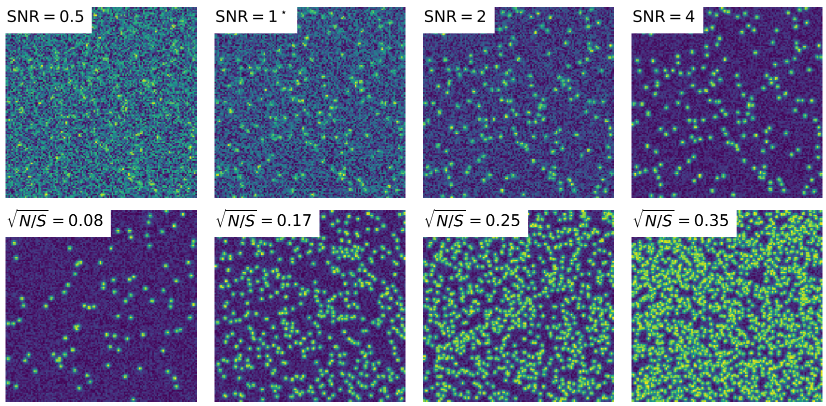


Figure 2: Effect of Signal to Noise Ratio (top) and particle number density on the generated images.

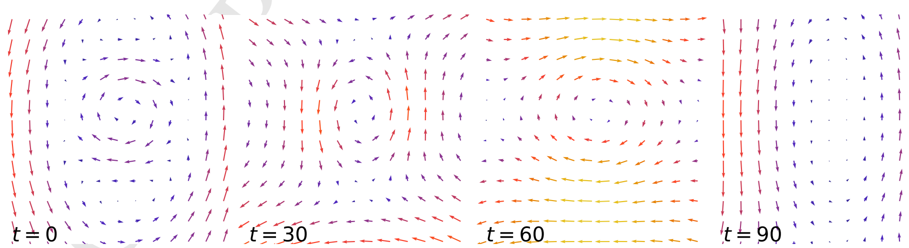


Figure 3: Space dependence of the synthetic vortex flow considered (Eq. 8) at various time instants.

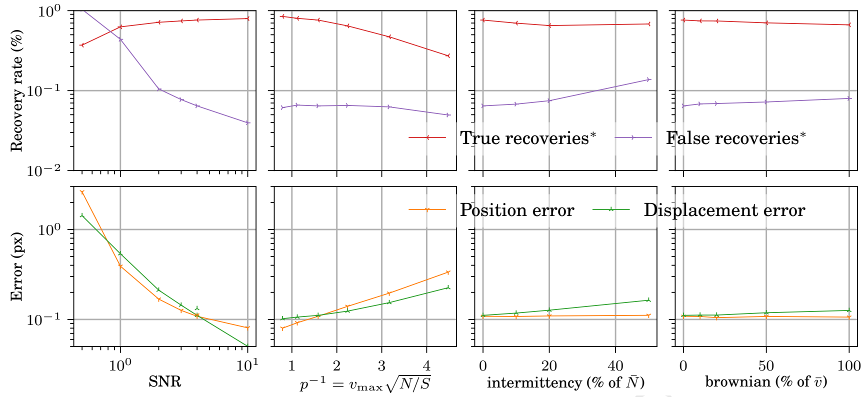


Figure 4: Algorithm accuracy function of 4 variables: Signal to Noise Ratio (SNR), object separation compared to mean displacement ($p^{-1} = \overline{v}\sqrt{N/S}$, with $\overline{v}$ the mean displacement and N the number of objects in the area S), object appearance and disappearance ("Intermittency") and velocities random fluctuations ("Brownian"). True recoveries rates are defined according to a maximum position error of d and a maximum displacement error of $v/2$. False recoveries are found for the opposite criterion. Default parameters are: SNR=4, N=1000, Intermittency=0 and Brownian=0.

4.1.2 Accuracy

Accuracy was measured owing to 4 indexes: mean percentage of true and false object detection as well as mean absolute error in object position and displacement estimation. These indexes were computed for various image qualities and flow properties. To better isolate TracTrac performances, no pre-processing step was performed on the synthetic images (e.g. background subtraction, median filter). First, TracTrac accuracy is compared to the Signal to Noise Ratio (SNR, defined as blob peak magnitude over magnitude of an underlying uniform noise). Results presented in Fig. 4 show that increasing SNR significantly increases tracking quality: for SNR≥ 4 (a typical value in PIV experiments), less than 5% of false detections are made, while mean position and displacement error are below 0.2 pixels, a value comparable with recent PTV methods [7]. Another quality factor is given by the ratio of maximum displacement length to the mean distance between neighbour objects expressed as $p^{-1} = \overline{v_{\max}}/\sqrt{N/S}$ (the inverse of the ratio defined by [13]). PTV algorithm are usually limited to $r \ll 1$ to avoid object association ambiguities between frames [13, 17]. Thanks to the motion predictor, the association rule, and the outlier filter, false detections remain below 6% for ratios $p^{-1} \approx 4.5$, while position and displacement error are below 0.5 pixels. To the author knowledge, such large values of p^{-1} have not yet been reported in the literature.

Appearance and disappearance of objects through time, referred to as "intermittency" in Fig. 4, often occur due to out of transverse velocities in 3D flows observed on 2D planes. While this phenomenon complicates the association process, the number of false tracklets remains limited to 12% at high intermittency levels (50% of the object

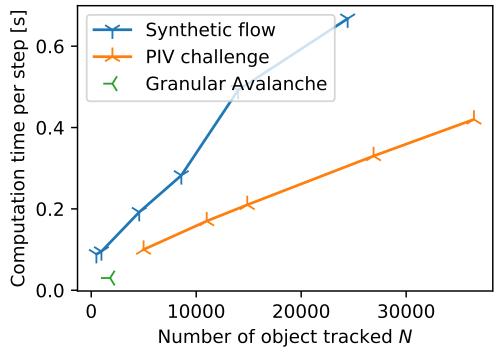


Figure 5: Computation time per video frame depending on the number of objects tracked. Computations are made with the Python implementation of TracTrac on a HP ELiteBook 840 laptop with processor Intel i7.

disappearing at each frame), suggesting a good adaptation of the algorithm to out of plane motions. While intermittency does not affect position error, it increases slightly the mean displacement error (owing to false associations, with mean displacement errors smaller than 0.5 pixel for a level of intermittency of 50%).

The last factor considered is the stochasticity of the underlying flow field, which cannot be predicted by deterministic motion predictors [5]. To investigate this effect, white noise was summed to object velocities in proportion of the deterministic flow velocity magnitude. Fig. 4 shows that both false detection and displacement error remain limited for fluctuations levels comparable with the average magnitudes (9% and 0.3 pixel respectively). This good performance is ensured by the continuous and local monitoring of prediction errors, that allows the computation of a local threshold to filter outliers, threshold which is directly influenced by the local motion statistics.

4.1.3 Efficiency

As shown by Fig. 5, TracTrac algorithm provides a computational time that grows only linearly with the number of object to track. This is mainly due to the implementation of k-d tree structures for nearest neighbour search. This allows for 25000 objects to be tracked in less than 0.7 seconds per frame (Fig. 5).

5 Application to geosciences

In this section, TracTrac specificities are highlighted through 3 examples of particular interest to geoscientists. We provide a comparison of TracTrac results with another open-source PIV software, PIVlab [24]. The latter was parametrized to compute velocity fields on 16×16 pixel interrogation windows.

5.1 Turbulent flow

The first example concerns 1000 frames of a turbulent duct flow past a series of hills, similar to aquatic bedforms or aeolian dunes (Fig.6a). The data was presented in the 4-th PIV challenge as an example of time dependent flow with strong velocity gradients and out of plane motions (intermittency) [9]. It is available online at www.pivchallenge.org/pivchallenge4.html. The flow is visualized through a 2-D laser sheet which illuminates seeded particles in a plane. TracTrac processing on such data can be appreciated in Fig. 6b and in the supplementary video online. This test case provides an example of the algorithm capabilities to compute time-average Eulerian flow quantities within high resolution (average flow magnitude and turbulent kinetic energy are presented in Fig.6c). In the dark regions where no object could be detected, the average values are kept empty (e.g., blank pixels in the right size of Fig.6c). The accuracy of the algorithm is specifically demonstrated by Fig. 6d where the streamwise time-average velocity profile close to the above wall is plotted. The profile closely follows the expected logarithmic law of the wall over several measurement points, and allows deducing the value of the local wall shear velocity ($u_* \approx 0.035$ m/s). While comparable to the TracTrac values in the bulk flow, the PIVlab-computed time-average velocity profile do not allow a clear identification of the inertial layer where the log-law applies. This is caused by the filtering effect imposed by the interrogation windows which bias velocity gradients close to the wall boundary. Wall boundary layer typically present a linear increase of the shear stress while approaching the wall, which permits deducing the shear velocity independently of the log-law of the wall. In the inertial layer, the total shear stress $\tau = u_*^2\rho$ (ρ is the water density) is approximated by the turbulent stresses $-\rho\left\langle u'v'\right\rangle$ (viscous stresses $\nu\rho\partial_y\langle u\rangle$ are negligible outside of the viscous layer, $\nu = 10^{-6}\mathrm{m}^2\mathrm{s}^{-1}$ being the kinematic viscosity of water). Extrapolation of the Reynolds stresses at the wall thus provides an estimation of the shear velocity $u_* = \sqrt{\tau/\rho}$. Fig. 6d shows that TracTrac predicts a similar wall shear velocity by this method, confirming its ability to measure precisely all the contributing scales of turbulence. In contrast, the Reynolds stresses predicted by PIVlab, while qualitatively similar, are much smaller than TracTrac values. This is once again an effect of the low-pass filter imposed by interrogation windows. This analysis is confirmed by a comparison of the root mean squared (RMS) streamwise velocity profile at $x = 100$px with the average of several PIV software presented in [9]. Fig. 7 shows that TracTrac RMS are significantly higher than typically measured by traditional PIV software, including PIVlab.

Finally, it is worth pointing that the sub-pixel resolution of TracTrac algorithm also enables the observation of the viscous sub-layer in the mean velocity profile (Fig. 6d, at $yu_*/\nu < 30$). The latter has a theoretical size of $\nu/u_* \approx 28\mu$m, which corresponds to 0.15 pixel in the images and can thus, in theory, be visualized by TracTrac.

Computational time for the hill test case were reported in Fig. 5. In this figure, the level of the object detection threshold was sequentially decreased to artificially increase the number of detected object and confirm the quasi linear dependence of the computational time on object number. PIVlab takes about 2 seconds to compute 15741 velocity vectors at each time step, where TracTrac takes less than 0.5 seconds to provide the double of vectors. A factor 8 is thus observed between computation

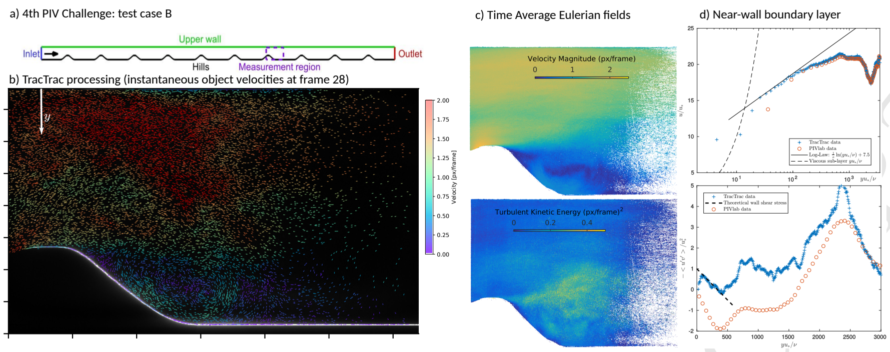


Figure 6: TracTrac results on the 4-th PIV Challenge data of time resolved turbulent flow past a hill [9]. (a) Geometry of the flow. (b) Instantaneous object velocities obtained by TracTrac. c) Time average Eulerian field: velocity magnitude (up) and turbulent kinetic energy (down). (d) Turbulent profiles close to the wall above the hill: mean streamwise velocity (up) and Reynolds stresses (down)

time of PIVlab and TracTrac.

5.2 Granular avalanche

The second example focuses on the avalanche of granular material (glass beads of 1mm diameter) along an inclined plate confined between two lateral walls (Fig. 8a). The experiment was made at Institut de Physique de Rennes, France, as part of a larger project aiming at modelling the rheology of dense inertial flow of granular media [8]. The purpose of this example is to highlight the role of the motion predictor step and the associated monitoring of prediction errors to resolve locally heterogenous flow regions. In this experiment, the image density of objects is about 0.13 object/pixels, with displacements up to 6 pixels/frames, giving locally a ratio $p^{-1} = \bar{v}/\sqrt{N/S} \approx 0.8$. Instantaneous top and side views of the granular flow are shown on Fig. 8b with a color scale proportional to the monitored error between motion prediction and corrected displacement, showing local variations in the error values. As beads are generally bouncing against the walls, these regions present higher deviations from the mean motion than the bulk of the flow. This is confirmed by transverse and vertical profiles (Fig. 8c) that show higher average prediction errors on the side walls and at the bottom of the plate (at $z = 160$px) than in the bulk of the flow. This increase is also observed in the mean kinetic agitation (defined here as $\sqrt{u'^2 + v'^2}$).

By continuously monitoring the local mean prediction error, the algorithm genuinely adapts to the Brownian nature of object motion close to the side walls. As a consequence, the threshold for outlier filtering (see Sec. 3.1.3), $\bar{\epsilon} + 1.5\sigma_\epsilon$, locally adapts to the flow characteristics and allows for an correct estimation of object motion statistics in all regions.

In contrast, PIVlab underestimates the kinetic agitation of the flow close to the

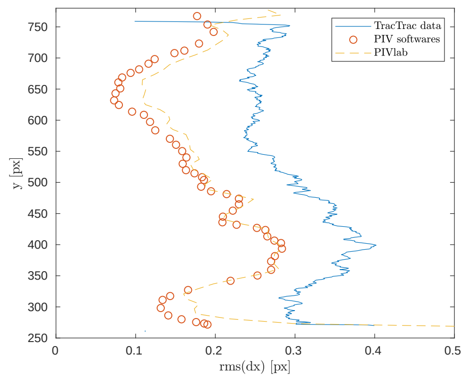


Figure 7: Root mean square streamwise velocity profiles estimated at the transverse section $x = 100$px of the hill test case. The average of five PIV algorithms (Dantec, DLR, INSEAN, IOT and IPP) are represented with circles (adapted from the Fig. 21 of [9]), together with the PIVlab estimates (dashed line) and TracTrac values (blue line).

side walls (Fig. 8c). An advantage of PTV over PIV also appears in the low density gaseous region that develops above the dense granular flow in the bottom view (for $x = 0$ to 75px). In this region, the kinetic agitation estimated by PIV increases artificially because interrogation windows are often empty, leading to erroneous velocity estimates. This effect is not occurring in TracTrac, since velocity is computed in a Lagrangian basis only where objects are detected.

5.3 Bird flock

In the last example, the fly of a bird flock recorded by Attanasi et al. [2] is used to highlight the versatility of the algorithm and its robustness for many types of motions (Fig. 9). In this example, bird motion is three-dimensional so that, in the image, bird trajectories can occlude each other. However, TracTrac rules out fake connections when ambiguity arises in the nearest neighbour association, producing sure tracklets. These tracklets can then be recombined with cost optimization algorithm to reconstruct each individual entire trajectory.

Another aspect well highlighted by this example is the equal ability of a single size, isotropic convolution kernel (here the differential of Gaussians) to predict the velocity of objects that are not always of isotropic neither Gaussian shape (the birds wings for instance). It is particularly true in videos where moving features are not particles as in the two first examples, but consist of a deforming texture (the water surface of a flowing river for instance). In these situations, an isotropic convolution will still be able to isolate local features of interest in the image; features which can be tracked to provide an estimation of local velocities. In general, it is enough for images to have strong, dense and aleatory intensity gradients to provide good features to track, and reliable tracking results.

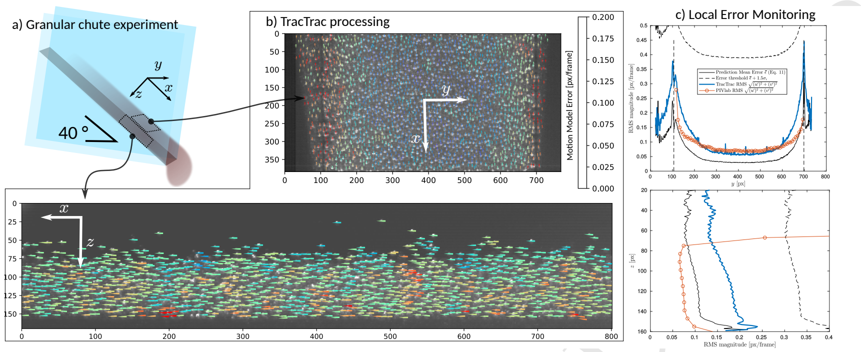


Figure 8: TracTrac genuine error monitoring revealed by a granular avalanche experiment.

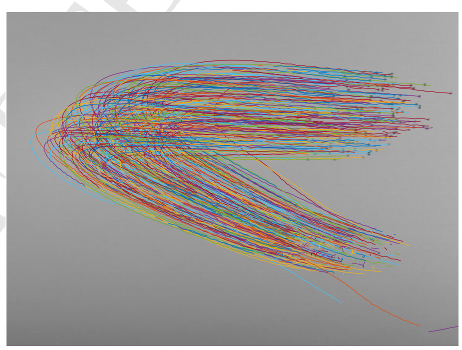


Figure 9: Birds trajectories obtained by TracTrac superimposed on the video of bird flock by Attanasi et al. [2]

6 Conclusion

In this article, I present an open source PTV algorithm called TracTrac, dedicated to motion recognition in geophysics. The main advantages of this algorithm are

1. A fast implementation through k-d tree nearest neighbour search, enabling the use of PTV for applications usually restricted to PIV.

2. An iterative prediction-correction procedure capable of following large object displacements in fluctuating and heterogeneous flow fields ($p^{-1} = 4.5$).

3. A robust 3-frame association process that limits velocity bias.

In particular, it has been shown that the algorithm provides much higher details of turbulent statistics than other open-source PIV software [24]. This result is crucial, since the measure of microscopic velocity fluctuations and sharp local gradients are often essential to correctly model geophysical processes (in turbulence and granular flows for instance).

All TracTrac source files are freely available (see Computer Code Availability section). Among the possible future developments, 3-dimensional tracking via stereoscopic videography may be easily implemented in the current algorithm. Other improvements such as the recognition of size and other specific features of objects can provide stronger constraints to the association process without increasing significantly the computation time.

Computer Code Availability

The TracTrac Matlab and Python source code are freely available at https://perso.univ-rennes1.fr/joris.heyman/tractrac-source.zip. Compiled versions are also available on SourceForge at https://sourceforge.net/projects/tractrac.

Acknowledgments

I am deeply acknowledging Alexandre Valance, Hervé Tabuteau, Renaud Delannay and Philippe Boltenhagen (Institut de Physique de Rennes) for funding the scientific project involving the granular chute experiment. I am also grateful to the 3 anonymous reviewers whose comments significantly contributed to improve the manuscipt.

References

[1] A. A. Aguirre-Pablo, M. K. Alarfaj, E. Q. Li, J. F. Hernandez-Sanchez, and S. T. Thoroddsen. Tomographic particle image velocimetry using smartphones and colored shadows. *Scientific Reports*, 7(1):3714, 2017.

[2] A. Attanasi, A. Cavagna, L. Del Castello, I. Giardina, T.S. Grigera, A. Jelic, S. Melillo, L. Parisi, O. Pohl, E. Shen, and M Viale. Information transfer and behavioural inertia in starling flocks. *Nature Physics*, 10:691, Jul 2014.

[3] B. Babenko, M. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, p983–990, Jun 2009.

[4] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, Sep 1975.

[5] N. Chenouard, I. Smal, F. de Chaumont, M. Maška, I.F. Sbalzarini, Y.o Gong, J. Cardinale, C. Carthel, S. Coraluppi, M. Winter, A. R Cohen, W. J Godinez, K. Rohr, Y. Kalaidzidis, L. Liang, J. Duncan, H. Shen, Y. Xu, K. E G Magnusson, J. Jaldén, H. M Blau, P. Paul-Gilloteaux, P. Roudot, C. Kervrann, F. Waharte, J.-Y. Tinevez, S. L Shorte, J. Willemse, K. Celler, G. P van Wezel, H.-W. Dan, Y.-S. Tsai, C. O. de Solórzano, J.-C. Olivo-Marin, and E. Meijering. Objective comparison of particle tracking methods. *Nature Methods*, 11:281, Jan 2014.

[6] K. T. Christensen. The influence of peak-locking errors on turbulence statistics computed from piv ensembles. *Experiments in Fluids*, 36(3):484–497, Mar 2004.

[7] C. Cierpka, B. Lütke, and C. J. Kähler. Higher order multi-frame particle tracking velocimetry. *Experiments in Fluids*, 54(5):1533, May 2013.

[8] J. Heyman, P. Boltenhagen, R. Delannay, and A. Valance. Experimental investigation of high speed granular flows down inclines. *EPJ Web Conf.*, 140:03057, 2017.

[9] C. J. Kähler, T. Astarita, P. P. Vlachos, J. Sakakibara, R. Hain, S. Discetti, R. La Foy, and C. Cierpka. Main results of the 4th international piv challenge. *Experiments in Fluids*, 57(6):97, May 2016.

[10] C. J. Kähler, S. Scharnowski, and C. Cierpka. On the resolution limit of digital particle image velocimetry. *Experiments in Fluids*, 52(6):1629–1639, Jun 2012.

[11] C J. Kähler, S. Scharnowski, and C. Cierpka. On the uncertainty of digital piv and ptv near walls. *Experiments in Fluids*, 52(6):1641–1656, Jun 2012.

[12] H. G. Maas, A. Gruen, and D. Papantoniou. Particle tracking velocimetry in three-dimensional flows. *Experiments in Fluids*, 15(2):133–146, Jul 1993.

[13] N. A. Malik, Th. Dracos, and D. A. Papantoniou. Particle tracking velocimetry in three-dimensional flows. *Experiments in Fluids*, 15(4):279–294, Sep 1993.

[14] J. C. McCall and M. M. Trivedi. Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation. *IEEE Transactions on Intelligent Transportation Systems*, 7(1):20–37, Mar 2006.

[15] D. Michaelis, D. R Neal, and B. Wieneke. Peak-locking reduction for particle image velocimetry. *Measurement Science and Technology*, 27(10):104005, 2016.

[16] H. Nobach, N. Damaschke, and C. Tropea. High-precision sub-pixel interpolation in particle image velocimetry image processing. *Experiments in Fluids*, 39(2):299–304, Aug 2005.

[17] K. Ohmi and H-Y. Li. Particle-tracking velocimetry with new algorithms. *Measurement Science and Technology*, 11(6):603–616, May 2000.

[18] N. T. Ouellette, H. Xu, and E. Bodenschatz. A quantitative study of three-dimensional lagrangian particle tracking algorithms. *Experiments in Fluids*, 40(2):301–313, Feb 2006.

[19] E.F.J. Overmars, N.G.W. Warncke, C. Poelma, and J. Westerweel. Bias errors in piv: the pixel locking effect revisited. In *15th Int Symp on Applications of Laser Techniques to Fluid Mechanics*, Lisbon, Portugal, 05-08 Jul 2010.

[20] T. Roesgen. Optimal subpixel interpolation in particle image velocimetry. *Experiments in Fluids*, 35(3):252–256, Sep 2003.

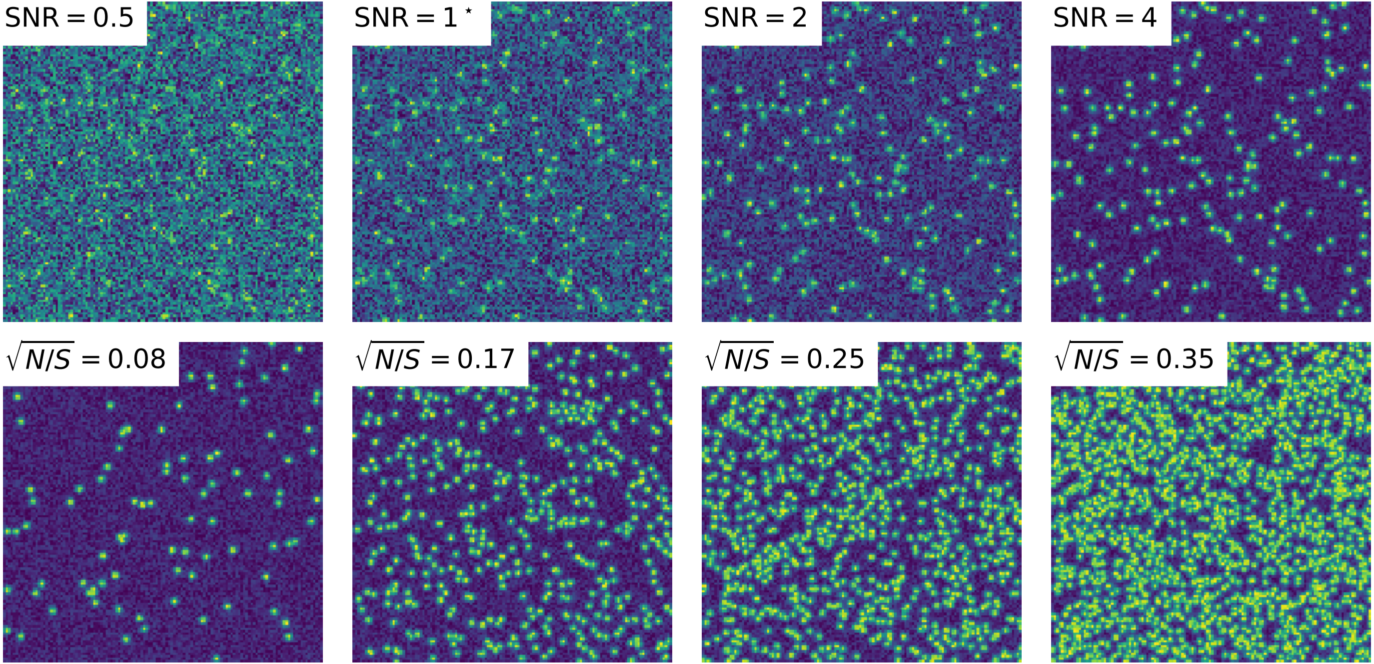
[21] D. Schanz, S. Gesemann, and A. Schröder. Shake-the-box: Lagrangian particle tracking at high particle image densities. *Experiments in Fluids*, 57(5):70, Apr 2016.

[22] J.F.G. Schneiders, I. Azijli, F. Scarano, and R.P. Dwight. Pouring time into space. In *11th International Symposium on Particle Image Velocimetry, PIV15*, Santa Barbara, CA, USA, 2015.

[23] I. Smal, M. Loog, W. Niessen, and E. Meijering. Quantitative comparison of spot detection methods in live-cell fluorescence microscopy imaging. In *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 1178–1181, Jun 2009.

[24] W. Thielicke and E. J. Stamhuis. PIVlab – towards user-friendly, affordable and accurate digital particle image velocimetry in MATLAB. *Journal of Open Research Software*, 2, Oct 2014.

[25] J. Westerweel, G. E. Elsinga, and R. J. Adrian. Particle image velocimetry for complex and turbulent flows. *Annual Review of Fluid Mechanics*, 45(1):409–436, 2013.


$\mathrm{SNR} = 0.5$
$\mathrm{SNR} = 1^{\star}$
$\mathrm{SNR} = 2$
$\mathrm{SNR} = 4$
$\sqrt{N/S} = 0.08$
$\sqrt{N/S} = 0.17$
$\sqrt{N/S} = 0.25$
$\sqrt{N/S} = 0.35$