15th International Symposium on Particle Image Velocimetry – ISPIV 2023
June 19–21, 2023, San Diego, California, USA

pyPTV - a complete framework to reconstruct physical flow fields from camera images

R. Barta[1,2,*], C. Bauer[1], C. Wagner[1,2]

[1] Institute of Aerodynamics and Flow Technology, German Aerospace Center (DLR), Germany

[2] Institute of Thermodynamics and Fluid Mechanics, Technical University of Ilmenau, Germany

[*] corresponding author: robin.barta@dlr.de

Abstract

We introduce a novel probability-based Particle Tracking Velocimetry (PTV) and Data Assimilation (DA) framework called pyPTV. It is an open-source project under continuous development and is completely written in the programming language Python. The framework provides a complete routine for processing measurement data suitable for PTV. Starting from raw camera images, pyPTV includes image processing, calibration, and a novel PTV algorithm to reconstruct individual seeding particle trajectories (tracks) in a Lagrangian frame of view. In addition, post-processing algorithms are implemented to refine the tracks. The framework includes interpolation and DA algorithms, that allow the user to generate physical flow fields in an Eulerian frame of view. The velocity field of the particle tracks is corrected to be solenoidal, and the pressure field is estimated. The implemented debugging scripts help fix problems with each subroutine during processing. To verify the method, a synthetic test case of turbulent Rayleigh-Bénard (RB) convection (Pr = 6.9, Ra = 1E10) in a cubic cell is generated by direct numerical simulation with massless tracer particles. A reconstruction accuracy of over 65% with 95% correct tracks is achieved at tracer particle densities of about 0.1 ppp.

1 pyPTV

The proposed framework includes the necessary tools to convert raw camera images (b16 or tif format) into particle tracks and to assimilate physical flow fields from the tracks. Specifically, pyPTV includes the following tools:

- image processing (section 1.1)
  (raw camera images $\rightarrow$ processed camera images),
- calibration (section 1.2)
  (raw calibration target images $\rightarrow$ Soloff calibration parameters (Soloff et al., 1997)),
- probability-based particle tracking velocimetry (section 1.3)
  (processed camera images & Soloff calibration parameters $\rightarrow$ tracks),
- post-processing (section 1.4)
  (tracks $\rightarrow$ refined tracks),
- interpolation (section 1.5)
  (refined tracks $\rightarrow$ Eulerian velocity fields),
- data assimilation (section 1.6)
  (Eulerian velocity fields $\rightarrow$ solenoidal Eulerian velocity fields and pressure fields).

The framework can be found on GitHub via the link provided in Barta et al. (2023). Its use is explained in a video, also available there, based on a new synthetic test case of turbulent RB convection in a cubic cell (see section 2). In the following subsections, we briefly discuss the main aspects of each algorithm. A full explanation can be found in Barta et al. (2023).

1.1 Image processing

The first tool of the framework helps to identify tracer particles and their center coordinate $\vec{x}^{(c_q)}(t) \in \mathbb{R}^2$ on camera images captured at time step t by camera c_q with $q \in [1, \cdots, N_c]$. The value N_c is the total number of cameras observing the flow. The images are processed by applying a mask generated by the tool itself. Next, all parts of the remaining image that do not change over time must be removed. This can be done using either a background subtraction method or a POD filter technique, as described in Mendez et al. (2017). Both of these methods are implemented in pyPTV. Then, an intensity threshold is applied, and the image is sharpened by first blurring it with a Gaussian kernel and then adding its weighted image. The remaining image is processed using a subpixel localization algorithm provided by the 'skimage.feature.peaklocalmax()' Python function. Finally, a 2D Gaussian fit is applied to the coordinates returned by the algorithm on the raw image to estimate the center coordinate based on the intensity distribution of the particle projection.

1.2 Calibration

A camera model is needed to recover the 3D position $\vec{X} \in \mathbb{R}^3$ of a tracer particle inside the measurement domain from camera images. PyPTV uses the nonlinear Soloff function with a parameter vector $\vec{a} \in \mathbb{R}^{19}$ defined by:

$$\begin{aligned} F_{\vec{a}} &: \mathbb{R}^3 \to \mathbb{R}, \\ F_{\vec{a}}(\vec{X}) &:= a_0 + X\left(X\left(a_9 X + a_{11} Y + a_{14} Z + a_4\right) + a_{13} Y Z + a_6 Y + a_7 Z + a_1\right) \\ &\quad + Y\left(Y\left(a_{12} X + a_{10} Y + a_{15} Z + a_5\right) + a_8 Z + a_2\right) \\ &\quad + Z\left(Z\left(a_{17} X + a_{18} Y + a_{16}\right) + a_3\right)). \end{aligned} \tag{1}$$

The Soloff camera model contains two Soloff functions, one for each dimension of the camera coordinate system. Two parameter vectors $\vec{a}_x^{(c_q)}, \vec{a}_y^{(c_q)} \in \mathbb{R}^{19}$ for each camera c_q are required to map a 3D position onto a camera image. Herzog et al. (2021) and Barta et al. (2023) proposed an iterative method that inverts the Soloff camera model to recover the 3D particle positions from 2D positions of multiple camera images. The idea behind a camera calibration is to estimate the parameter vectors for each camera observing the measurement domain. PyPTV provides a calibration in two steps. First, an initial calibration based on images of a marker target which is moved through the measurement domain at known positions. Second, a volume self calibration method designed for the Soloff camera function based on the idea of Wieneke (2008).

1.3 Probability-based particle tracking velocimetry

The main routine of the framework is the PTV algorithm 1.

Algorithm 1 PTV

```
 1: procedure
 2:     Input: processed images, Soloff parameters
 3:     for $t_n \in \{ t_{start}, \dots, t_{start} + N_{init} - 1 \}$ do
 4:         Triangulation: estimate 3D particle positions based on processed images at time step $t_n$
 5:     Initialisation: build tracks from the first $N_{init}$ 3D particle positions
 6:     for $t_n \in \{ t_{start} + N_{init}, \dots, t_{end} \}$ do
 7:         Predict: probability-based approximation of the next track position, velocity and acceleration
 8:         Tracking: if possible, extend all tracks with appropriate 3D particle positions
 9:         Delete: delete the extended particle positions from the processed images of time step $t_n$
10:         Triangulation: estimate 3D particle positions based on residual images at time step $t_n$
11:         Initialisation: build new tracks from the last unused $N_{init}$ 3D particle positions
12:         Smoothing: correction of track positions, velocities and accelerations to their most probable
    values
13:     return tracks
```

The triangulation subroutine combines epipolar geometry and the reprojection of possible candidates based on the Soloff camera model. The triangulation is estimated for each permutation of the camera orientation to process each combination of possible candidates. Duplications are then excluded and removed form the final list by defining a minimum distance between two 3D particles. The pruning vector idea introduced by Tan et al. (2020) is used to filter the resulting particle list so that each image coordinate is used only once in the triangulation process of 3D particle positions. Tracks are initialized via the histogram linking approach in combination with a cost function minimization over all possible candidates, see Herzog et al. (2021); Barta et al. (2023). The prediction, tracking and smoothing routine are all based on the novel probabilistic approximation method introduced by Barta et al. (2023). Each component of the track position $\vec{X}(t) = [X(t), Y(t), Z(t)]$, velocity $\vec{V}(t) = [V_X(t), V_Y(t), V_Z(t)]$, and acceleration $\vec{A}(t) = [A_X(t), A_Y(t), A_Z(t)]$ is approximated by a set of N Gaussian basis functions, e.g.:

$$X(t) = \sum_{i=1}^{N} \omega_i \exp\left(-\frac{(t-\mu_i)^2}{2\,\sigma_i^2}\right). \tag{2}$$

Each component has its own set of weights ω, means μ and standard deviations σ. A linear system is set up to solve for each of these parameters for each component of a given track. The inputs are the positions of a track and approximations of its velocity and acceleration. White noise with a standard deviation of twice the maximum position amplitude is added to each position input. The velocity and acceleration inputs are estimated from the noised position using the Savitzky-Golay filter, see Savitzky and Golay (1964). The linear system is solved multiple times for each track for different white noises applied to the position input. A probabilistic prediction of a track is made based on the ensemble average of the resulting weights, means, and standard deviations. Therefore, system parameters such as search radius and maximum allowed velocity and acceleration shifts are automatically determined for each track individually. A track is extended by the most likely 3D position where particles have been imaged in the vicinity of its projection onto the cameras. The efficiency of the probabilistic approximation method and its prediction of tracks is shown in figure 1.

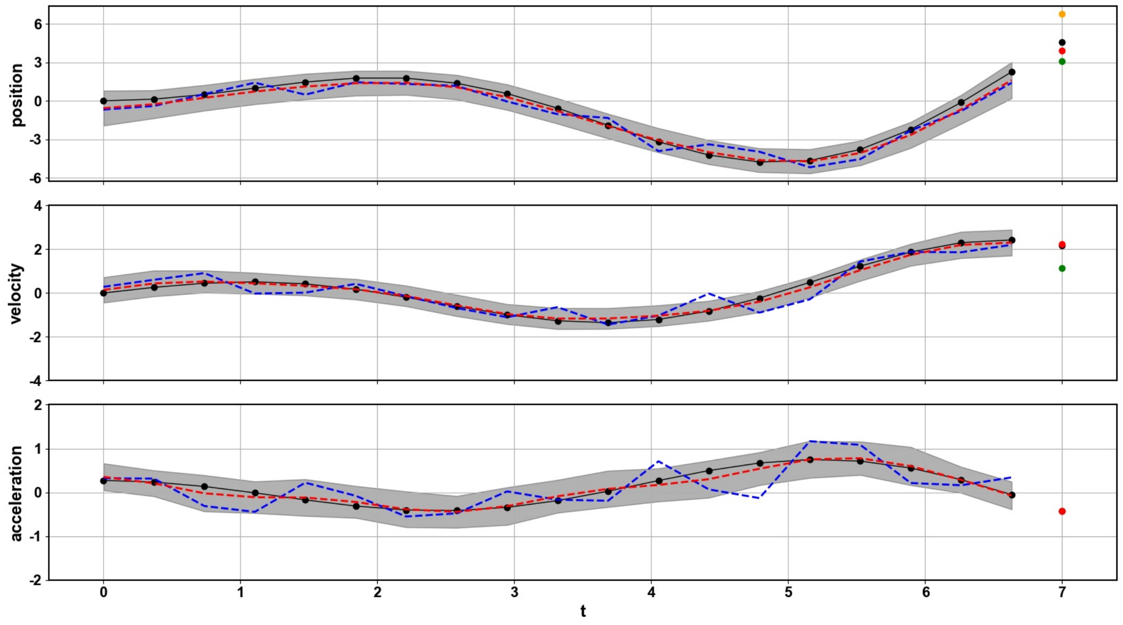

Figure 1: The function $X(t) = t\,\sin(t)$ (dotted black line) with $N_t = 20$ data points is noised (dashed blue line) and used as input for the probabilistic approximation. Here, $N = 12$ Gaussians are used to approximate the function. The resulting most probable track positions, velocities and accelerations (dashed red line) are shown with their corresponding error bands (gray tubes) at the top, middle and bottom respectively. The different coloured dots at $t = 7$ represent the prediction of the track values, where the orange dot corresponds to the prediction with a B-spline and the green dot with the Savitzky-Golay filter. The red dot shows the prediction based on the probabilistic approximation.

For a test we use the function for the position we use $X(t) = t \cdot sin(t)$. The function is noised with a small uniform noise of 20% of the maximum position amplitude. Especially for predicting the noised track, the method outperforms standard spline extrapolation methods like B-splines or the Savitzky-Golay filter (Schröder and Schanz, 2023). In addition, pyPTV provides a gap tracking method, that can extend a track over multiple time steps if the extension condition is not satisfied in previous time steps. The gap tracking method helps in reconstructing long and stable tracks.

1.4 Post processing

PyPTV offers two post processing methods to refine the tracks generated during the PTV processing. The first is the backtracking method, which runs the PTV algorithm again on each track but backwards in time. Due to the iterative nature of the PTV algorithm, new tracks are generated at each time step but since the PTV algorithm only extends the tracks forward in time, many tracks are missing their beginning. The backtracking algorithm helps to extend the length of the tracks. Second, the track repair algorithm can be used to join two tracks when the track of the same particle broke apart during the processing, e.g. due to missing image information or processing errors. These two methods help to refine the particle tracks and increase their accuracy for reconstructing the observed flow.

1.5 Interpolation

The Eulerian velocity vector fields $\vec{u}_m$ are interpolated from the Lagrangian velocity fields defined by the particle tracks using a radial basis function interpolation method, see Fasshauer (2007). This interpolation method is analytically smooth and thus achieves the estimation of spatial derivatives of the velocity vector field.

1.6 Data assimilation

Up to this point, the divergence-free correction of an interpolated velocity vector field $\vec{u}_m$ and the assimilation of a pressure field p are implemented. Both methods use the fractional step (Chorin, 1967, 1968) based on the Navier-Stokes equation (Navier, 1838) without forcing:

$$\frac{d\vec{u}}{dt} = \frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\,\vec{u} = -\nabla p + \nu \Delta \vec{u}, \tag{3}$$

$$\nabla \cdot \vec{u} = 0, \tag{4}$$

of an incompressible velocity vector field $\vec{u}$, where ν is the kinematic viscosity of the fluid and du/dt is the material derivative [1]. Note that the above equations are not valid for an interpolated velocity vector field $\vec{u}_m$ obtained from the PTV tracks since the latter carries noise induced by processing uncertainties during track generation and interpolation. Consequently, the fractional step needs to be applied two times at a certain time step t_n. First, to make the velocity vector field $\vec{u}_m$ incompressible and second, to assimilate the pressure field from the resulting incompressible velocity field. In detail, the fractional step algorithm is performed by splitting the partial time derivative in equation (3) into an approximate velocity vector field $\vec{u}^*$ at time step t^*. Rearranging terms yields two equations:

$$\frac{\vec{u}^* - \vec{u}_n}{t^* - t_n} = \Delta \vec{u}_n - (\vec{u}_n \cdot \nabla)\,\vec{u}_n, \tag{5}$$

$$\frac{\vec{u}_{n+1} - \vec{u}^*}{t_{n+1} - t^*} = -\nabla p_n, \tag{6}$$

[1]The material derivative is estimated directly from the acceleration $\vec{A}$ of the particle tracks. In general, there is no need to estimate the partial time derivative of two successive flow fields and add the convection term to estimate the material derivative. PTV estimates this term directly at each time step. This applies to the advancement of data assimilation techniques in modeling physical flow fields governed by the Navier-Stokes equations based on PTV data, as there is no need to interpolate the flow fields into an Eulerian frame of reference.

which sum up to the time-discrete Navier-Stokes equation. Here, index n indicates each property at time step t_n. Note that the incompressibility condition holds true:

$$\nabla \cdot \vec{u}_{n+1} = \nabla \cdot \vec{u}_n = 0.$$

Applying the divergence operator to equation (6) yields a Poisson equation for the pressure p_n:

$$\nabla \cdot \left(\frac{\vec{u}_{n+1} - \vec{u}^*}{t_{n+1} - t^*} \right) = \nabla \cdot (-\nabla p_n) \quad \Rightarrow \quad \frac{\nabla \cdot \vec{u}^*}{t_{n+1} - t^*} = \Delta p_n. \tag{7}$$

To make the interpolated velocity vector field $\vec{u}_m$ divergence free, two steps are required. First, Poisson equation (7) is solved implicitly on a staggered grid (Piller and Stalio, 2004) with Neumann boundary conditions by invoking $\vec{u}^* \to \vec{u}_m$ and a pseudo-pressure field $p_n \to \tilde{p}$. We then compute the solenoidal velocity vector field using the Helmholtz-Hodge decomposition (Bhatia et al., 2012):

$$\vec{u}_{sol} = \vec{u}_m - (t_{n+1} - t^*) \nabla \tilde{p}. \tag{8}$$

Finally, by invoking $\vec{u}_n \to \vec{u}_{sol}$ in equation (5), a velocity vector field $\vec{u}^*$ is estimated. Solving the Poisson equation (7) again but with $\vec{u}^*$ the pressure field p_n inside the Navier-Stokes equation at time step t_n is determined.

2 Synthetic test case

PyPTV comes with a new synthetic test case that is also available for free. It involves a direct numerical simulation (DNS) of a turbulent RB convection in a cubic cell with Rayleigh number Ra =1E10 and Prandtl number Pr = 6.9. The DNS is based on the experimental RB convection cell by Schiepel et al. (2013), which has similar system parameters. The side length of the cubic cell is $H = 500\,\mathrm{mm}$ and the temperature difference between heating and cooling plate is $\Delta T = 6\,\mathrm{K}$. Massless tracer particles are added to the flow generated by the DNS in order to simulate tracer particles. The test case is useful for validating PTV and DA algorithms because it provides correct physical informations about the position, velocity, acceleration, pressure, and temperature values of each individual particle. Details of the DNS can be found in Barta et al. (2023). The considered DNS solves the dimensionless transport equations for mass, momentum, and temperature using the Boussinesq approximation assuming that the fluid is incompressible:

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla) \vec{u} = -\nabla p + \sqrt{\mathrm{Pr}/\mathrm{Ra}}\, \Delta \vec{u} + T \vec{e}_3, \tag{9}$$

$$\frac{\partial T}{\partial t} + \vec{u} \cdot \nabla T = \sqrt{1/(\mathrm{Pr}\,\mathrm{Ra})}\, \Delta T, \tag{10}$$

$$\nabla \cdot \vec{u} = 0. \tag{11}$$

Here, $\vec{u} = (u_X, u_Y, u_Z)$ is the velocity vector field, p the pressure field, T the temperature field, and $\vec{e}_3$ the unit vector in vertical Z-direction. Equations (9)-(11) govern the problem of RB convection in a rectangular box with a heated bottom plate ($T = T_w$), a cooled top plate ($T = T_c$) and adiabatic side walls depicted in figure 2. Each velocity component in equations (9)-(11) has been made dimensionless with the free-fall velocity $\hat{u}_{ref} = (\lambda g \Delta T H)^{1/2}$, the spatial coordinates with the cell height $\hat{X}_{ref} = H$, the time coordinate with the corresponding reference time $\hat{t}_{ref} = \hat{X}_{ref}/\hat{u}_{ref}$ and the pressure with the reference pressure $\hat{p}_{ref} = \rho \hat{u}_{ref}^2$, where ρ is the fluid density. The temperature is non-dimensionalized by $\hat{T}_{ref} = \Delta T$ with $\Delta T = T_w - T_c$ and then shifted by $T_0 = (T_w - T_c)/(2\Delta T)$ to values between -0.5 and 0.5. No-slip and impermeability boundary conditions are applied to all walls. In addition, the top and bottom plate are modelled isothermally, while the side walls are modeled adiabatically. Furthermore, the dimensionless equations (9)-(11) are discretized spatially with a fourth-order accurate finite volume scheme detailed in Kaczorowski and Wagner (2009).

In time the equations are discretized with a second-order accurate Euler-leapfrog scheme. According to Wagner et al. (1994), the temporal discretization of equation (9) for a Leapfrog time step yields

$$\frac{1}{2\Delta t}\left(\vec{u}_{n+1}-\vec{u}_{n-1}\right)+\left(\vec{u}_{n}\cdot\nabla\right)\vec{u}_{n}=-\nabla p_{n}+\sqrt{\mathrm{Pr}/\mathrm{Ra}}\,\Delta\vec{u}_{n-1}+T_{n}\vec{e}_{3}, \tag{12}$$

where index n indicates the current discrete time step t_n. Δt denotes the time increment between two time steps. To integrate equation (12) in time, the fractional step introduced by Chorin (1967, 1968) is applied.

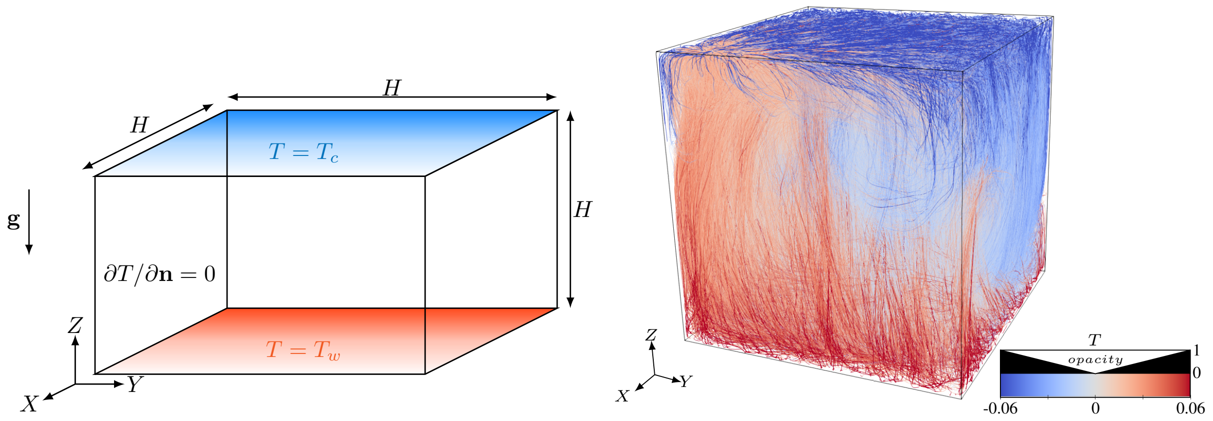

Figure 2: *Left:* cubic RB convection cell with volume H^3 and H being the cell height. All walls are no-slip boundaries; top and bottom walls are iso-thermal and side walls are adiabatic. *Right:* particle tracks from DNS coloured with the corresponding particle temperature. $N_{DNS} = 64000$ particles.

After an initial transient, when the simulation reaches a statistically stationary state ($t = t_0$) $N_{DNS} =$ 64000, massless tracer particles are homogeneously seeded and tracked in physical time for about 20 convective free fall times $\hat{t}_{ref}$. About 2/3 turnovers of the large-scale circulation (LSC) can be observed Sakievich et al. (2016). The LSC corresponds to a metastable dynamics typical for RB convections in closed domains, see Brown and Ahlers (2006). Figure 2 shows all particle trajectories extracted from the DNS. A rapid mixing of the initially homogeneous particle distribution is observed due to the intense turbulent fluctuations that develop for this high Rayleigh number. The trajectories are coloured with the corresponding particle temperature. The LSC is clearly visible as a warm upward motion in the left corner and a cold downward motion in the right corner of the cell shown in figure 2.

2.1 Image generation

Camera images are used as input for the pyPTV framework. They capture the motion of the tracer particles from the DNS (section 2) at discrete time steps. The synthetic test case used to generate the results uses $N_c = 4$ cameras. These cameras observe the flow from one of the four sidewalls of the cubic measurement domain, see figure 2. The resulting domain from the DNS is scaled by a factor of $H = 500mm$ to reproduce scales similar to the experiment mentioned in 2. Each camera image has a resolution of 800×800 pixels and a 16-bit grayscale channel. N_p particles with 3D positions $\vec{X}$ are mapped to image coordinates $\vec{x}$=(x_c,y_c) using the Soloff camera model. Each particle is initialized with an intensity value I randomly chosen between $I_1 = 1500$ and $I_2 = 2500$. The intensity distribution of the projected particle is estimated via a normalized Gaussian distribution with standard deviation of 1 pixel and the intensity I as amplitude. The imaged particle covers a 3×3 pixel wide area around the projection center (x_c,y_c). About 68% of the total intensity are located on the images. Camera noise is added to each pixel and it is uniformly randomized between 0 and the current pixel intensity divided by 20, resulting in a signal-to-noise ratio of 20 : 1. The synthetic images are generated at a given recording frequency f, measured in Hz. The physical time scales of the DNS are recovered before the images are generated. The intensity fluctuations from one time step to another are mimicked by multiplying a uniform distribution, which allows an intensity shift of maximum 20%. If the intensity distributions of several projected particles overlap at a pixel, the value of the brightest intensity is assigned. The recording frequency f and the number of particles can be adjusted to generate test cases of increasing levels of difficulty. The studied cases range from 0.02 ppp for the lowest number of particles ($N_p = 16000$) up to a particle density of about 0.1 ppp for the highest number of particles ($N_p = 64000$). At 8 Hz recording frequency the particles move about 6 pixels between each time step on the camera images. For 4 Hz this shift is doubled.

2.2 Validation

Figure 3 shows the results of processing different test cases with pyPTV without post processing for 50 time steps. For details on the parameters see Barta et al. (2023). Each reconstructed track is compared to the ground truth information of the particles seeded in the DNS. If both the particle location and its connection to the next particle are correct compared to the ground truth particles, the track is considered as correct. Figure 3 left visualizes the mean percentage of reconstructed particles that match with the ground truth are visualised for each test case with $N_p \in \{16000, 32000, 48000, 64000\}$ particles and $f \in \{4, 8\}$ Hz recording frequency. Figure 3 right shows the corresponding percentage of correct tracks.

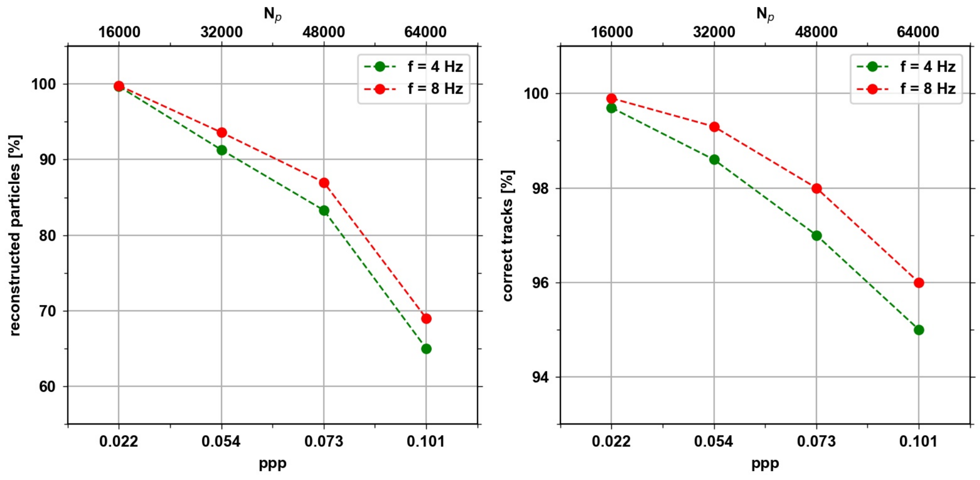

Figure 3: Left: the percentage of track points that match the ground truth information. Right: the percentage of all tracks that are considered correct because their assigned identification number persists over time. The data for two different recording frequencies: $f = 8$ Hz (red) and $f = 4$ Hz (green), is presented.

Figure 4 shows all tracks reconstructed while processing the case with $N_p = 16000$ particles and 4 Hz recording frequency. The color coding with the vertical velocity component V_z shows the LSC along a cell diagonal (LSC diagonal). The tracks are processed with the DA routine. Figure 5 compares the resulting solenoidal Euler velocity vector field and the assimilated pressure field along the LSC diagonal and its off-diagonal with the corresponding fields from the DNS at a given time step. The corrected velocity vector fields are similar to the DNS fields in their values, shape and reconstructed structures. Differences can barley spotted with naked eye. The assimilated pressure field is undetermined by a constant but its shape is similar to the pressure field extracted from the DNS. However, both pressure fields have the same minima within a vortex of the velocity vector field and higher values at the corners of the cell. Along the LSC off-diagonal the pressure field is more similar to the DNS field than along the LSC diagonal. The main difference comes from neglecting the force term in the fractional step, see section 1.6, which is $T\vec{e}_3$ in the case of RB convection.

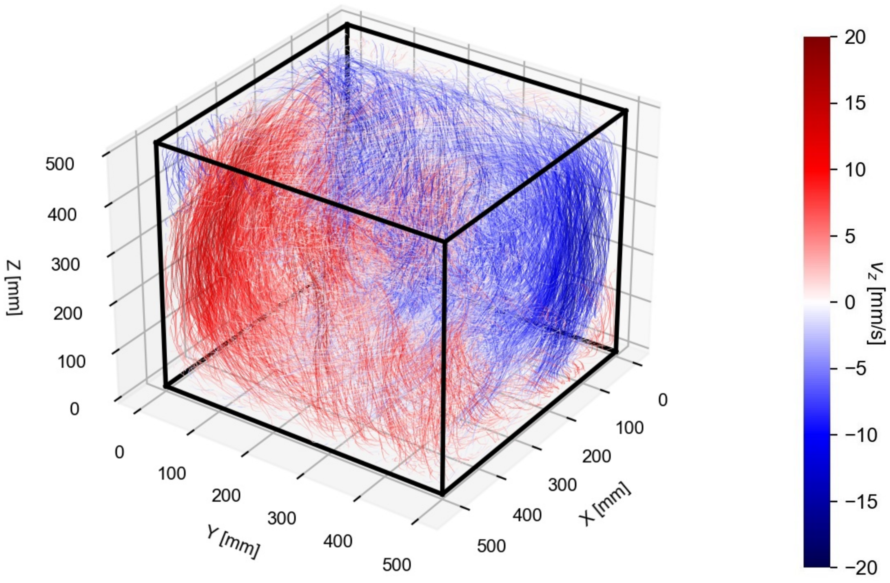

Figure 4: All tracks generated with pyPTV while processing the synthetic test case with parameters $N_p = 16000$ and $f = 4\,\text{Hz}$. The tracks are coloured according to their vertical velocity component V_z.

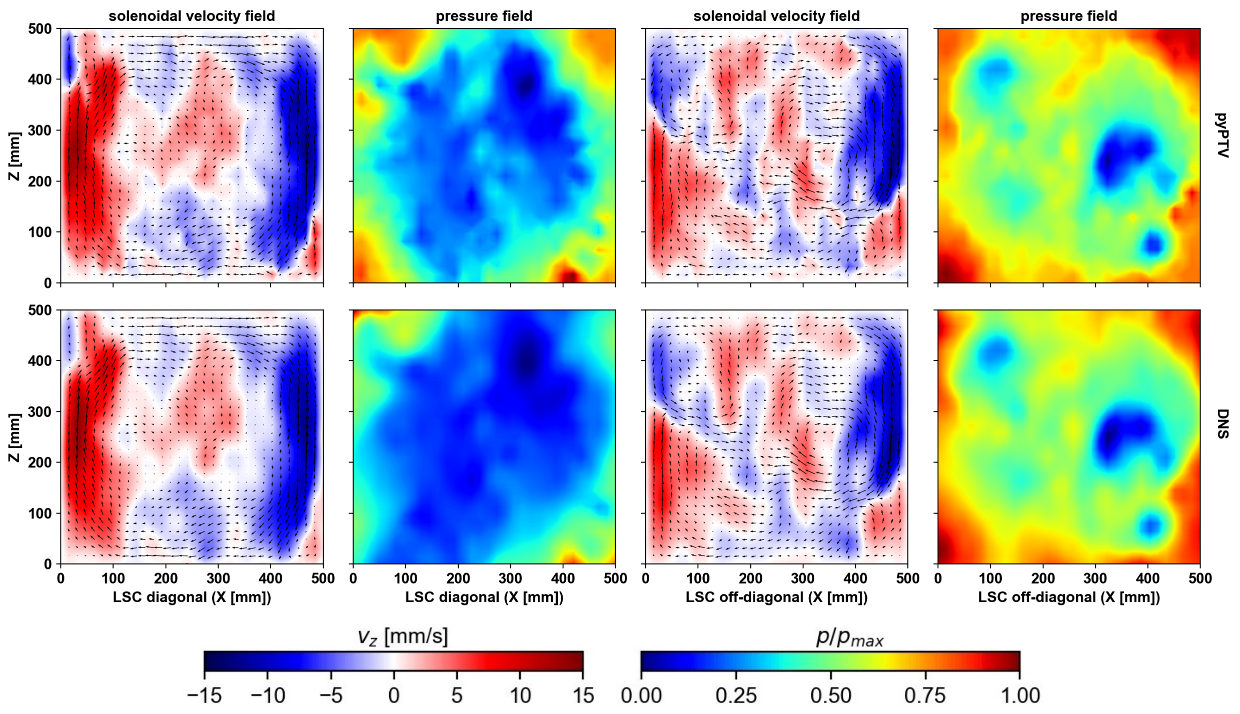

Figure 5: Assimilated solenoidal velocity and pressure fields based on the tracks generated from the synthetic test case with parameters $N_p = 16000$ and $f = 4\,\text{Hz}$. The fields are visualized along the LSC diagonal and its off-diagonal with a resultion of 32×32 data points. All velocity vector fields are coloured according to their vertical velocity component V_z. The assimilated pressure field is undetermined by a constant. Therefore, the pressure is coloured by its normalized value for each plot.

3 Conclusion

An overview of the pyPTV framework is given. Its performance in reconstructing particle trajectories via PTV and assimilating physical flow fields from the tracks is validated using a synthetic turbulent RB convection case generated using DNS.

In figure 3 the tracks generated by pyPTV are compared with the ground truth information from the DNS for multiple test of generated synthetic images by varying the recoding frequency and the number of tracer particles. As the number of particles is increaded, the number of reconstructed particles varies from nearly 100% for 16000 particles (0.02 ppp) to 65% for 64000 particles (0.1 ppp) for a recording frequency of 4 Hz. The results are similar for the recording frequency of 8 Hz. The percentage of correct tracks based on the reconstructed particles varies from nearly 100% for 16000 particles (0.02 ppp) to 95% for 64000 particles (0.1 ppp) for a recording frequency of 4 Hz. Again similar values are obtained at a recording frequency of 8 Hz.

The tracks are used to test the data assimilation routine. A solenoidal velocity vector field is estimated from the interpolated velocity vector field and used to assimilate a pressure field based on the approach discussed in 1.6. Figure 5 compares the results of the assimilation with the velocity and pressure fields extracted from the DNS at the same spatial resolution. Both fields are compared along the LSC diagonal and off-diagonal. The assimilated fields are similar to the DNS fields. The data assimilation routine of pyPTV is a first setup for future work. It can be seen as a playground to start DA research based on mesh-free or mesh-based assimilation methods, see Bauer et al. (2022).

Future work will test pyPTV and its features on various test cases such as experimental data. PyPTV is developed with the goal of making PTV more accessible and easier to use for individual purposes. The novel framework includes all tools needed to process raw camera images up to the assimilation of physical flow field. Further improvement and collaboration within the PTV community is of our interest to advance this area of research.

References

Barta R, Bauer C, Herzog S, Schiepel D, and Wagner C (2023) pyPTV: A comprehensive framework for probability-based particle tracking velocimetry and data assimilation. *Journal of Computational Physics* (under review)

Bauer C, Schiepel D, and Wagner C (2022) Assimilation and extension of particle image velocimetry data of turbulent Rayleigh–Bénard convection using direct numerical simulations. *Experiments in Fluids* 63:1–17. DOI https://doi.org/10.1007/s00348-021-03369-3

Bhatia H, Norgard G, Pascucci V, and Bremer PT (2012) The helmholtz-hodge decomposition: A survey. *IEEE Transactions on Visualization and Computer Graphics* 19:1386–1404. DOI https://doi.org/10.1109/TVCG.2012.316

Brown E and Ahlers G (2006) Rotations and cessations of the large-scale circulation in turbulent Rayleigh-Bénard convection. *Journal of Fluid Mechanics* 568:351–386. DOI https://doi.org/10.1017/S0022112006002540

Chorin AJ (1967) A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics* 2:12–26. DOI https://doi.org/10.1016/0021-9991(67)90037-X

Chorin AJ (1968) Numerical solution of the Navier-Stokes equations. *Mathematics of Computation* 22:745–762. DOI https://doi.org/10.2307/2004575

Fasshauer GE (2007) *Meshfree approximation methods with MATLAB*. volume 6. World Scientific

Herzog S, Schiepel D, Guido I, Barta R, and Wagner C (2021) A probabilistic particle tracking framework for guided and brownian motion systems with high particle densities. *SN Computer Science* 2:1–20. DOI https://doi.org/10.1007/s42979-021-00879-z

Kaczorowski M and Wagner C (2009) Analysis of the thermal plumes in turbulent Rayleigh–Bénard convection based on well-resolved numerical simulations. *Journal of Fluid Mechanics* 618:89–112. DOI https://doi.org/10.1017/S0022112008003947

Mendez M, Raiola M, Masullo A, Discetti S, Ianiro A, Theunissen R, and Buchlin JM (2017) POD-based background removal for particle image velocimetry. *Experimental Thermal and Fluid Science* 80:181–192. DOI https://doi.org/10.1016/j.expthermflusci.2016.08.021

Navier CL (1838) Navier Stokes equation. *Chez Carilian-Goeury (Paris)*

Piller M and Stalio E (2004) Finite-volume compact schemes on staggered grids. *Journal of Computational Physics* 197:299–340. DOI https://doi.org/10.1016/j.jcp.2003.10.037

Sakievich P, Peet Y, and Adrian R (2016) Large-scale thermal motions of turbulent Rayleigh-Bénard convection in a wide aspect-ratio cylindrical domain. *International Journal of Heat and Fluid Flow* 61:183–196. DOI https://doi.org/10.1016/j.ijheatfluidflow.2016.04.011

Savitzky A and Golay MJE (1964) Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry* 36:1627–1639. DOI https://doi.org/10.1021/ac60214a047

Schiepel D, Bosbach J, and Wagner C (2013) Tomographic particle image velocimetry of turbulent Rayleigh-Bénard convection in a cubic sample. *Journal of Flow Visualization and Image Processing* 20. DOI https://doi.org/10.1615/JFlowVisImageProc.2014010441

Schröder A and Schanz D (2023) 3d Lagrangian particle tracking in fluid mechanics. *Annual Review of Fluid Mechanics* 55. DOI https://doi.org/10.1146/annurev-fluid-031822-041721

Soloff SM, Adrian RJ, and Liu ZC (1997) Distortion compensation for generalized stereoscopic particle image velocimetry. *Measurement Science and Technology* 8. DOI https://doi.org/10.1088/0957-0233/8/12/008

Tan S, Salibindla A, Masuk AUM, and Ni R (2020) Introducing openlpt: new method of removing ghost particles and high-concentration particle shadow tracking. *Experiments in Fluids* 61:1–16. DOI https://doi.org/10.1007/s00348-019-2875-2

Wagner C, Friedrich R, and Narayanan R (1994) Comments on the numerical investigation of Rayleigh and Marangoni convection in a vertical circular cylinder. *Physics of Fluids* 6:1425–1433. DOI https://doi.org/10.1063/1.868257

Wieneke B (2008) Volume self-calibration for 3d particle image velocimetry. *Experiments in Fluids* 45:549–556. DOI https://doi.org/10.1007/s00348-008-0521-5