User friendly cross-platform 3D particle tracking velocimetry

Renewable Energy & Turbulent Environment Group
Department of Mechanical Science and Engineering
University of Illinois at Urbana-Champaign

Experiments at Mouline

Particle types:

Plastic spheres

Beans

Grass

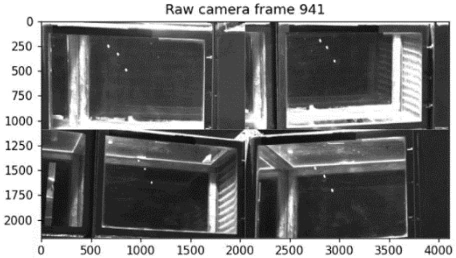
3D printed plastic rectangular piece

Cases:

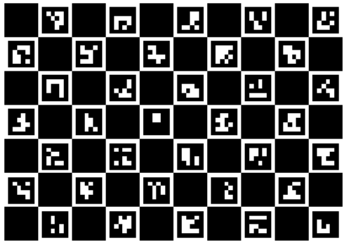
- Single particle
- Three particle
- Many particles (50 ~ 200)

PTV mode:

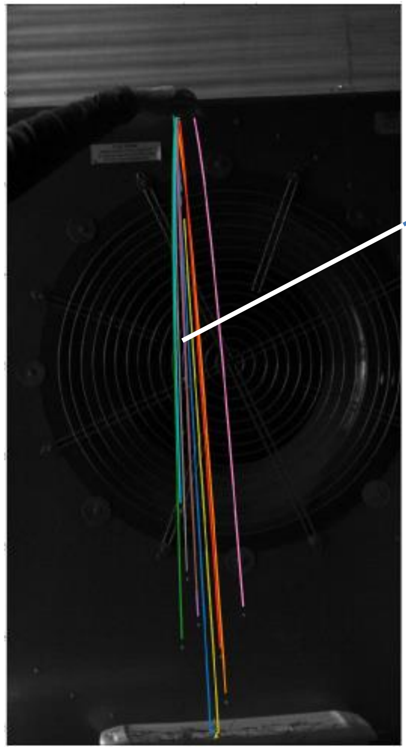
- 3D PTV with 4 cameras
- 2D PTV with single camera



Calibration target: Charuco board

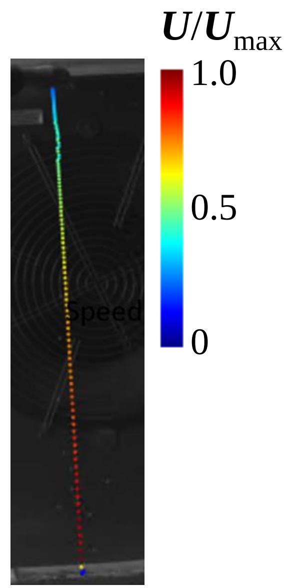


2D PTV of free fall particles



Particle
trajectories

Color indicates different
particles



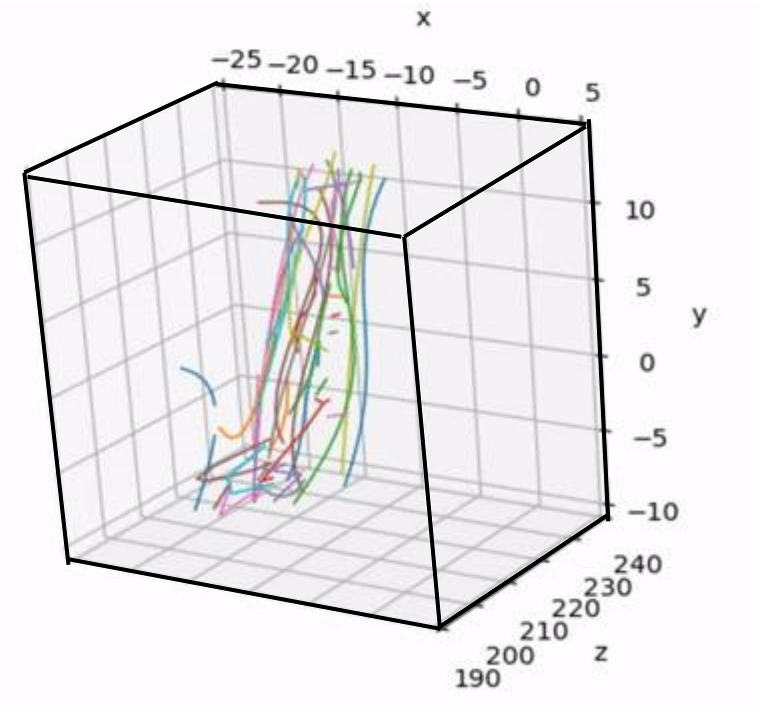
Speed, $\boldsymbol{U}$

$\boldsymbol{a/g} = 1$

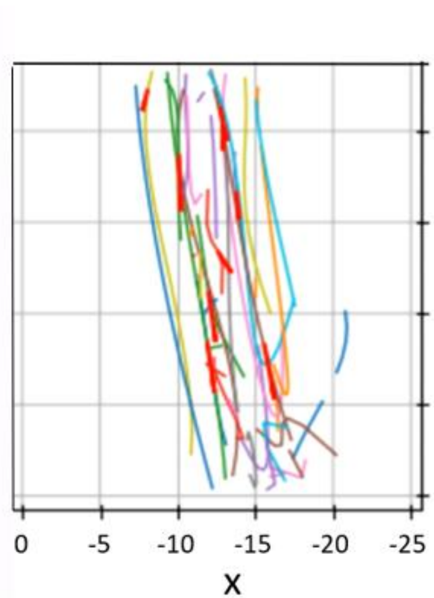
Acceleration, $\boldsymbol{a}$

Example of a selected trajectory.
Velocity and acceleration.

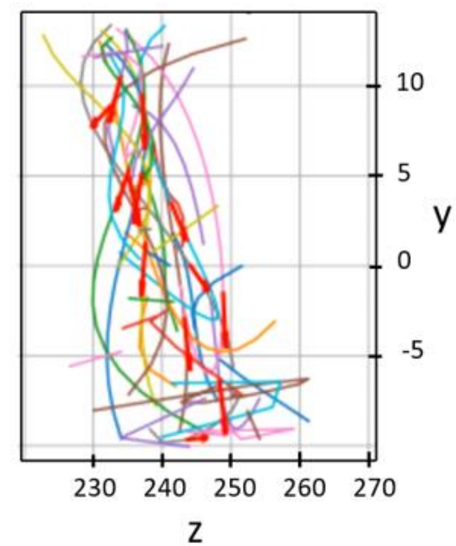
3D PTV for spherical target tracking



3D view



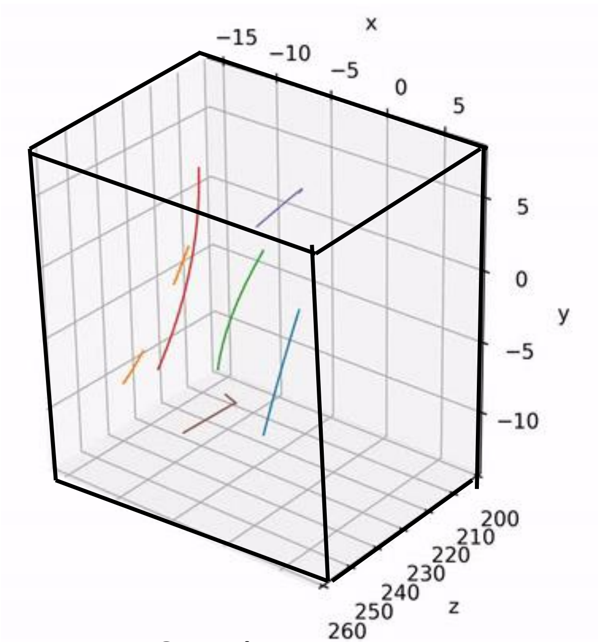
x-y view



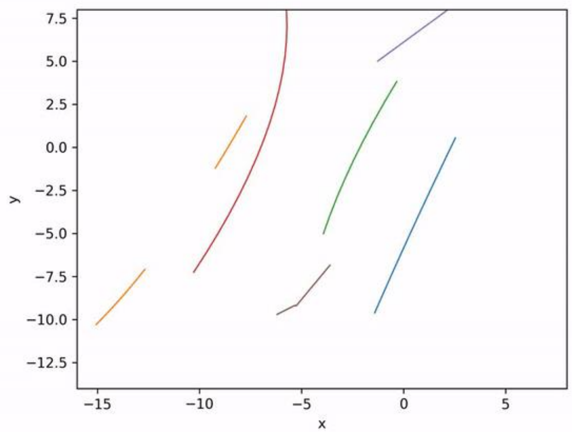
y-z view

The red vector, which indicates the velocity of particles, is visible in that frame.
The right part of the video displays the 2D-2D particle matching for the corresponding frames.

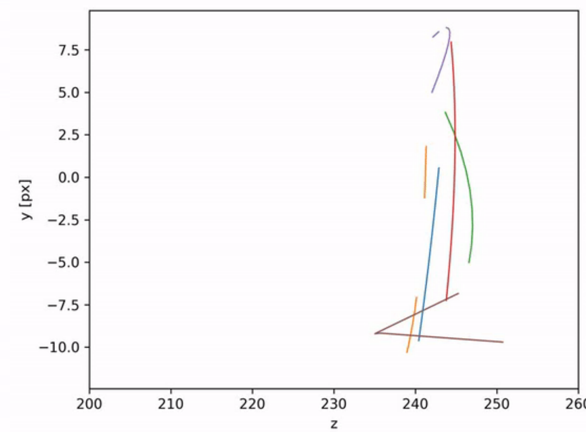
3D PTV for 3D printed plastic rectangular piece



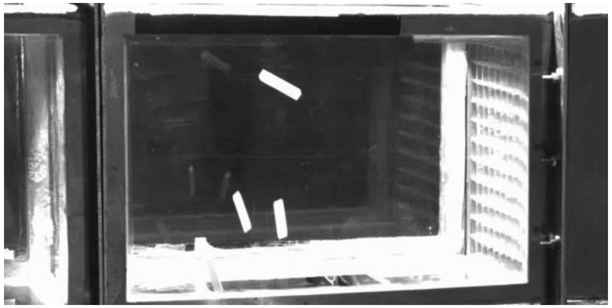
3D view



x-y view



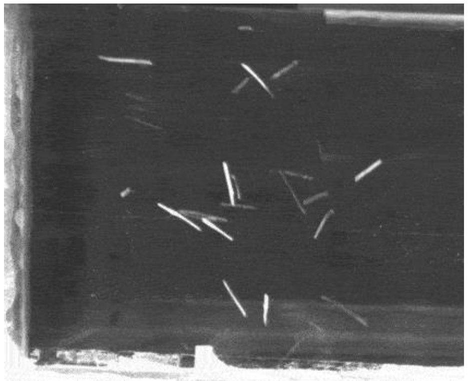
y-z view



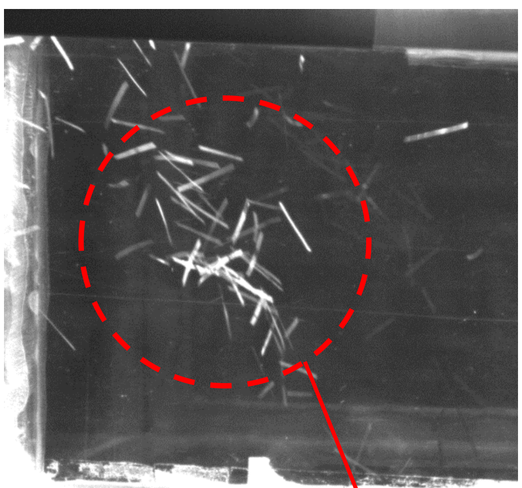
4 pieces tracking

Because the target has a rectangular shape, it appears relatively large compared to the small investigation volume, high density making it difficult to distinguish and match them from different views. The targets' projections are overlapping, and we are currently using line detection to separate them (maybe we shouldn’t mention)

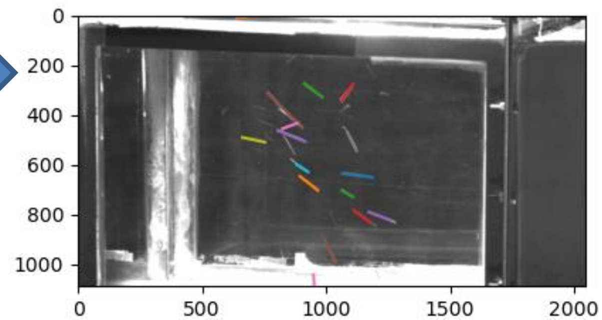
3D PTV for grass



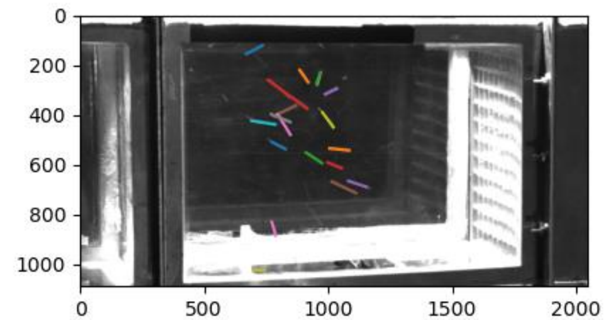
Maximum density we can track



Too dense to track

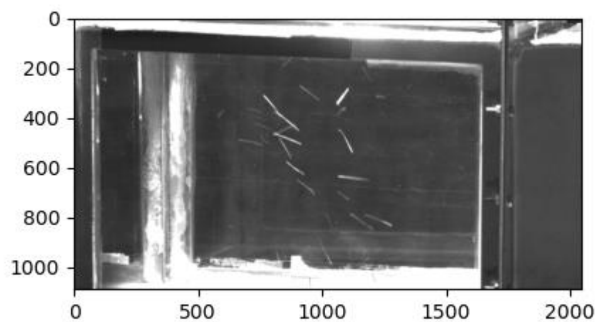


Grass detected view 1

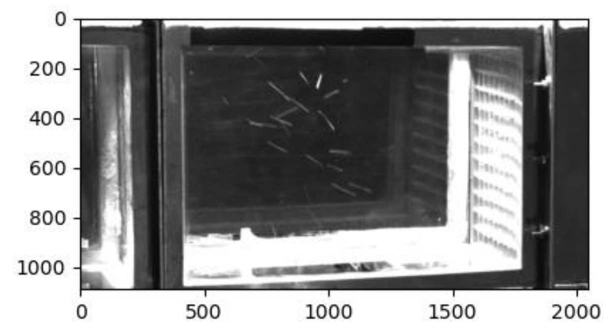


Grass detected view 2

…



View 1

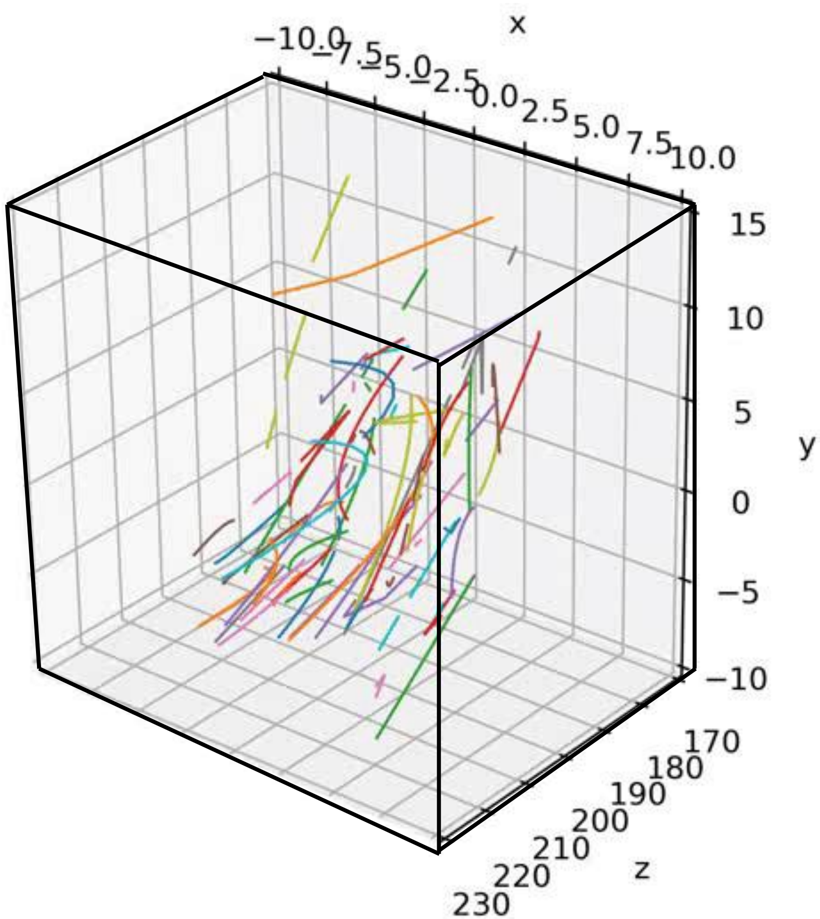


View 2

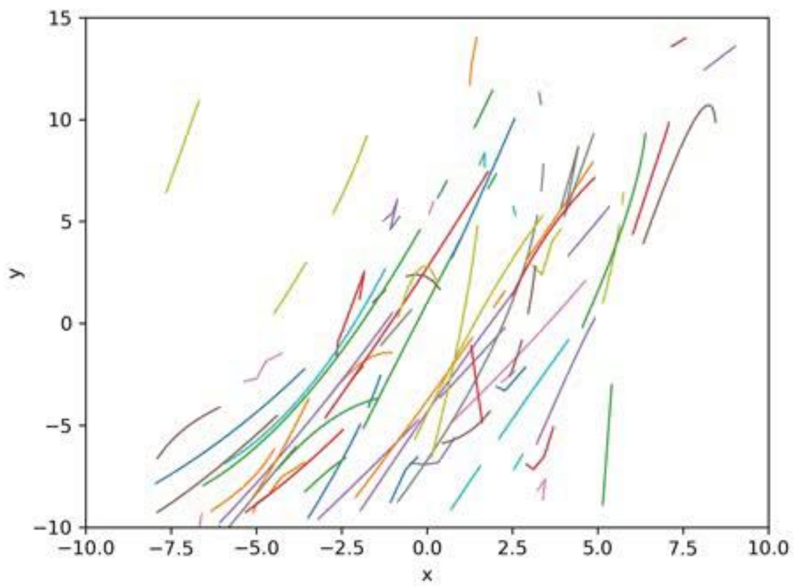
…

- OSTU threshold method used for preprocessing.
- Hough transform used for edge detection.
- K cluster mean method to label each grass.

3D PTV for grass

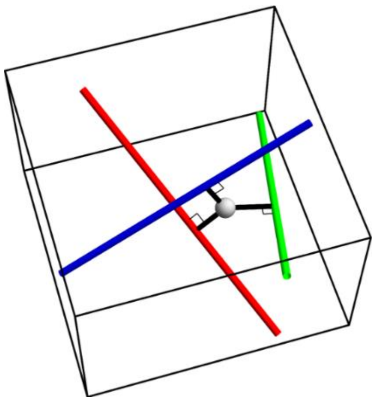


3D view

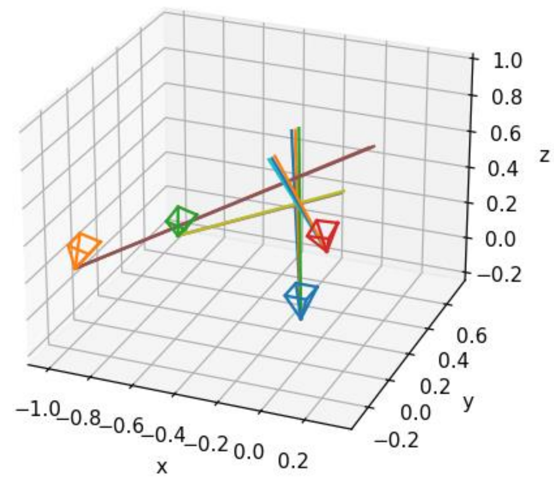


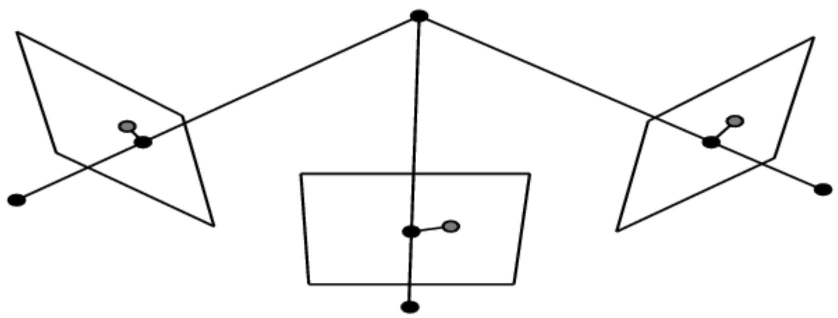
x-y view

Large particle in small volume



- DLT algorithm(we previously used) is that it can be sensitive to variations, calibration, unsynchronized camera and error of center detection.
- Relays on converge of 4 rays, which mean small delay in one of 4 camera can destroy the tracking, especially when FOV is small.





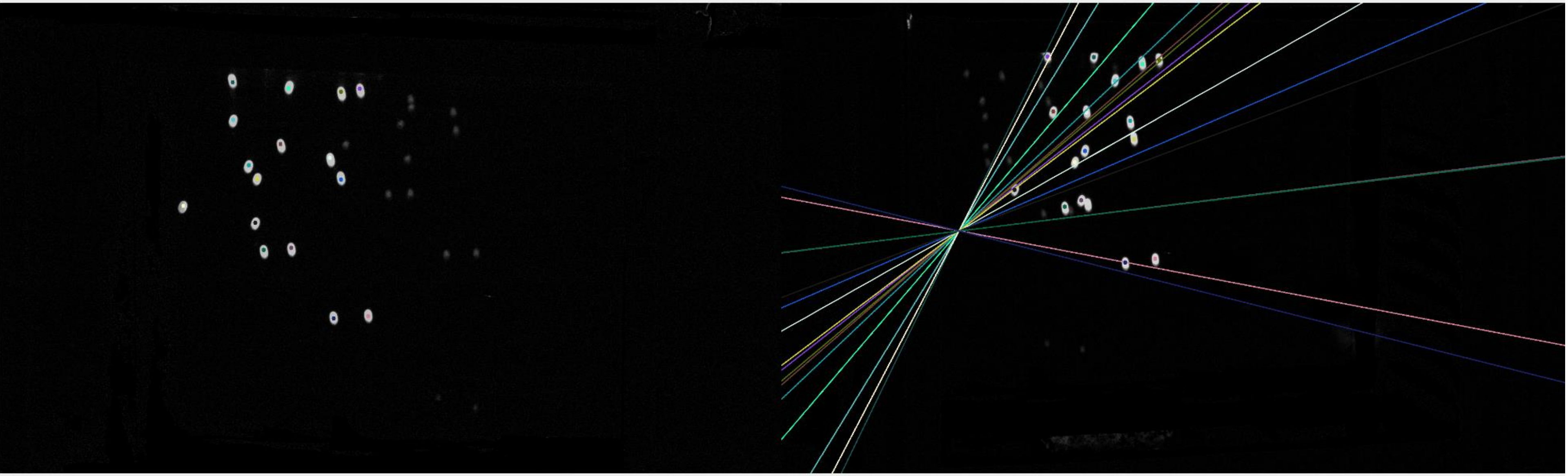
No delay example

Delay appears in Mouline**’s** experiment and large particle in small volume tracking is uncommon for PTV. How to make the system more robust under error from delays and center detection?

Modified epipolar searching mixed with homography finder

1. Generate epipolar line corresponding to particle in camera 1.
2. Search along epipolar line in camera 2,3,4.
3. Since there is delay, sometimes the particle won't be very close to epipolar line.
4. Use epipolar line combined with homography finder method to search for potential particle.

Although this function has been migrated from openptv, simply using epipolar searching is insufficient to match all points across views due to minor delays.



Modified epipolar searching mixed with homography finder

1. Regard the particle in 3D position as a rigid body.
2. Since delay the rigid body size and shape is slightly changed and modified.
3. Iterative Closest Point algorithm used to fine tune the coordinates to match 2d particle projection.
4. Calculate homography mapping matrix between each pair of camera.
5. Run coarse mapping with ransac filter.
6. Run dense mapping based on homography mapping matrixing.

User interface design: Overview

- 5 steps

1. Load package needed for script [...]

2. Create Charuco Board as calibration target [...]

TODO: create ui to make it easier to create charucoboard (done)

3. Single Camera calibration [...]

4. Multi Camera calibration [...]

5. 2D particle (Seed) tracking [...]

TODO: 3 of 4 camera detection updating (section 5.5) (done)

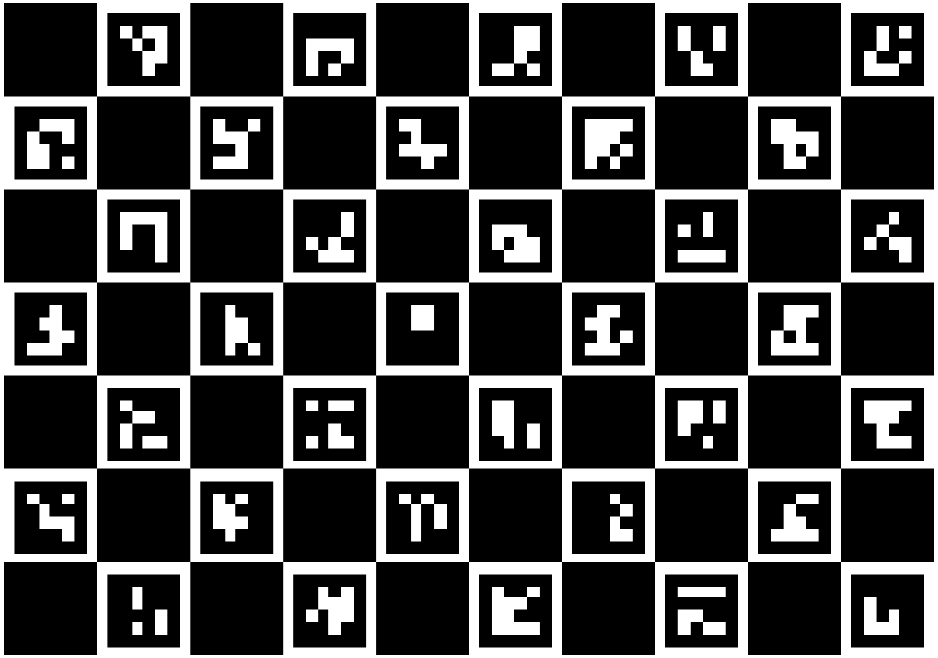
- Step1: simple import wrapped program

```
from LiuHongTracking import LiuHongPTV
```

User interface design: step 2

- Create calibration board & print it

```
PTV = LiuHongPTV()
PTV.calibrationTarget(7,10, False)
```



User interface design: step 3

- Calibration each lens parameter: focal length, distortion and so on.
- Folder selection UI for easy calibration folder selection.

Please choose the image folder contains camera 1 camera calibration images:

Select No selection

Current selected camera 1 path : /media/liu/My Book/data_back/calibration_camera1

Please choose the image folder contains camera 2 camera calibration images:

Select No selection

Current selected camera 2 path : /media/liu/My Book/data_back/calibration_camera2

Please choose the image folder contains camera 3 camera calibration images:

Select No selection

Current selected camera 3 path : /media/liu/My Book/data_back/calibration_camera3

Please choose the image folder contains camera 4 camera calibration images:

Select No selection

Current selected camera 4 path : /media/liu/My Book/data_back/calibration_camera4

User interface design: step 3

- 3 lines code easy lens calibration.
- Calibration result automatic backup for repeated use.

3.3 Overview and pre prune of single camera calibration images

Note : Usaually use 30 fps to take the image sequence for each camera as calibration images. The overall number of images for each camera calibration needed should be around 100~500, which ensure calibration accurancy and efficiency. You will use the number of skipped frame to skip every N images to save the time.

In [29]:
```
PTV.match_images()
```

```
interactive(children=(IntSlider(value=10, description='Image_Skip', min=1), Output()), _dom_classes=('widget-i…
```

3.3.1 prune the image

In [28]:
```
PTV.prune_images()
```

3.4 Finding the Intrinsic Parameters

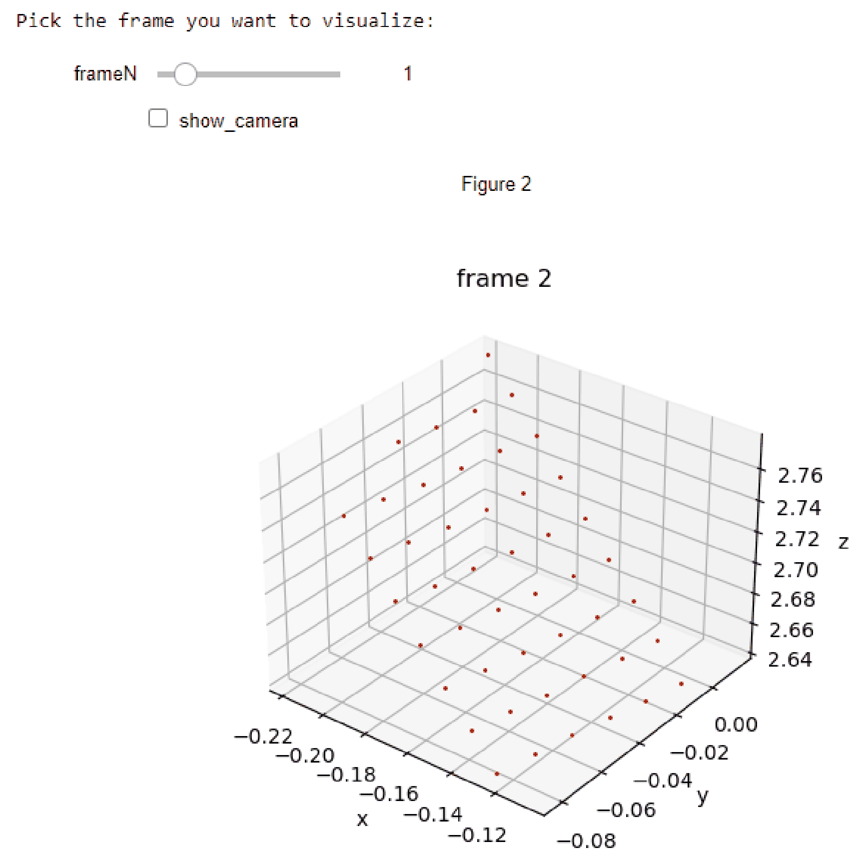
Note: May take 1-2 minutes to calibration each camera

In [8]:
```
PTV.single_calibration()
```

```
K1: [[9.40888847e+03 0.00000000e+00 8.72013887e+02]
 [0.00000000e+00 9.40888847e+03 5.80387566e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
reprojection error: 0.2813540915758296
K1: [[8.31823986e+03 0.00000000e+00 1.15409371e+03]
 [0.00000000e+00 8.31823986e+03 5.65714353e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
reprojection error: 0.3462537989212902
K1: [[8.70465693e+03 0.00000000e+00 8.98375847e+02]
 [0.00000000e+00 8.70465693e+03 5.68487421e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
reprojection error: 0.3342489427687012
K1: [[7.85750538e+03 0.00000000e+00 1.22174054e+03]
 [0.00000000e+00 7.85750538e+03 6.66007808e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
```

User interface design: step 4

- 5 lines code for fast multi camera calibration.
- User interface to visualize the camera calibration result with calibration board detection.



User interface design: step 5

- Data folder selection UI.

Please choose the image folder contains camera 1 particle images:

Select No selection

`Current selected camera 1 path :  /media/liu/My Book/data_back/experiment2_camera1`

Please choose the image folder contains camera 2 particle images:

Select No selection

`Current selected camera 2 path :  /media/liu/My Book/data_back/experiment2_camera2`

Please choose the image folder contains camera 3 particle images:

Select No selection

`Current selected camera 3 path :  /media/liu/My Book/data_back/experiment2_camera3`

Please choose the image folder contains camera 4 particle calibration images:

Select No selection

`Current selected camera 4 path :  /media/liu/My Book/data_back/experiment2_camera4`

User interface design: step 5

- Synchronization checker UI for name matching.

```
Please Pick the timestamps matched for all cameras
```

camera1	10-40-49
camera2	10-41-48
camera3	10-38-19
camera4	10-39-17

- Synchronization checker UI for image index matching: automatic prune image sequences

```
[23]: PTV.match_particle_images()
['10-45-49', '10-46-48', '10-44-21', '10-45-05']
Verify if the number of images are same for each cameras( warning: if not same, need to check the image sequece ):
```

	Camera1	Camera2	Camera3	Camera4
0	1671	1672	1666	1667

```
Warning: Some folder have different number of images
Checking if first images of all camera matched ... ...

Camera 1 first image name: 10-45-49.000-0001.tif
Camera 2 second image name: 10-46-48.000-0001.tif
Camera 3 third image name: 10-44-21.000-0001.tif
Camera 4 fourth image name: 10-45-05.000-0001.tif

Click the check box and continue if you think they are matched or closer enough
```

☐ matched

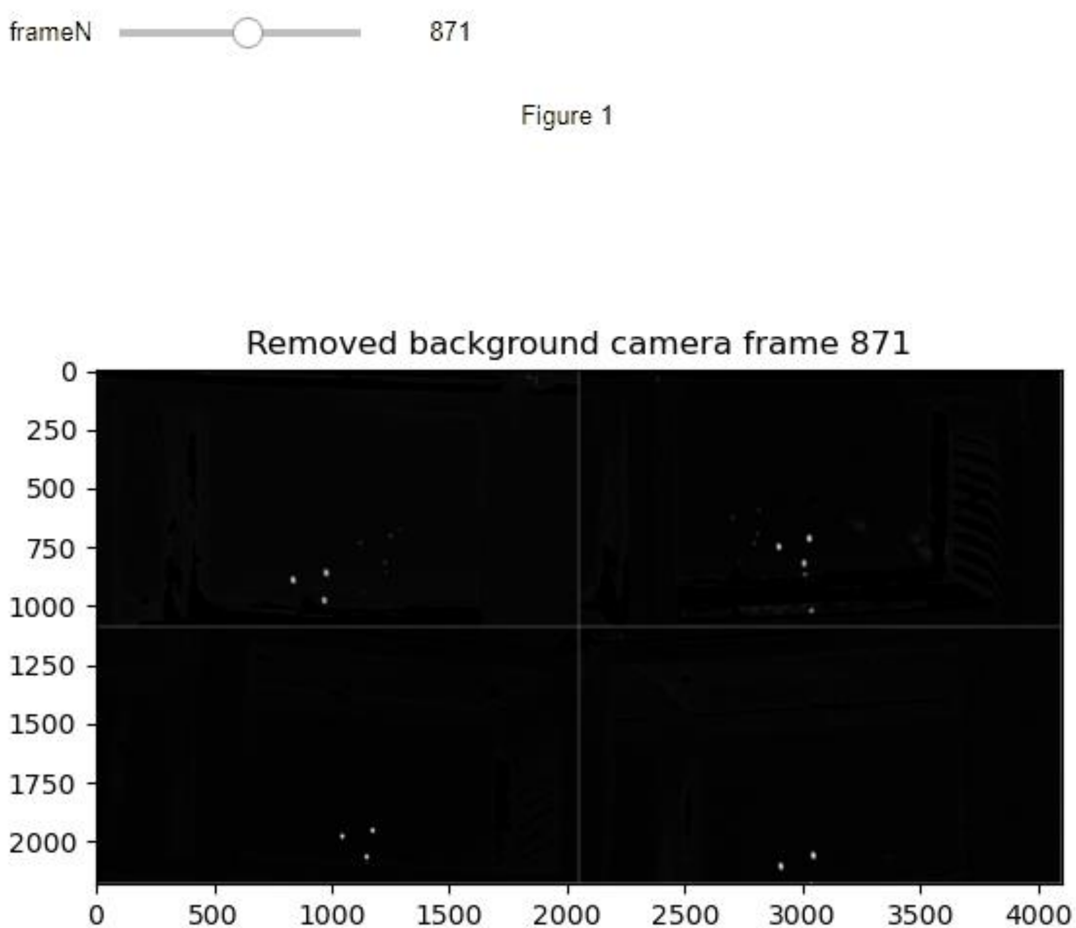
```
Error: Can not pair images for 4 cameras, please check the images folder
```

5.4 Visualize the raw matched images and remove the background

Note: In this section, user are required to pick single frame as background images, which is images without any particle. Background images are frames

User interface design: step 5

- Preprocessing: background remove UI



User interface design: step 5

- Preprocessing: particle detector tuning UI for 3 type of objects with hyperlink.

5.5 Find the particle center

Note: In this section user are required to tune the parameter to detect the particle center for 2D images. The particle size on each camera view should be similiar for processing

Type of object:

1. Sphere shape particle
2. grass (elongated particle)
3. Large plastice piece

User interface design: step 5

- Preprocessing: particle detector tuning UI for 3 type of objects with hyperlink.

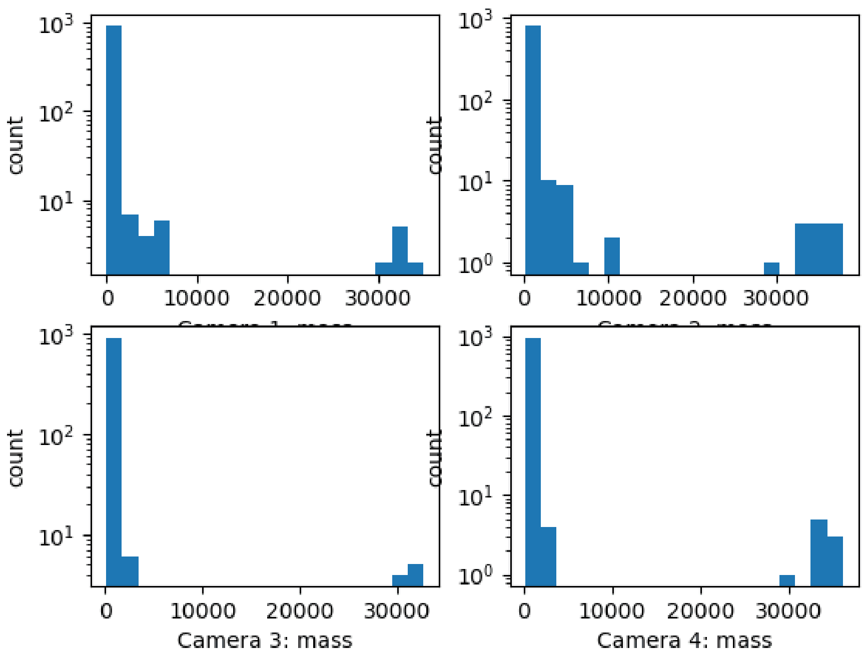
frameN 881

particle_size 27

```
Pick the minimum intensity of each particle (mass) :
Based on particle number on raw images, we can roughly decide the threshold
```

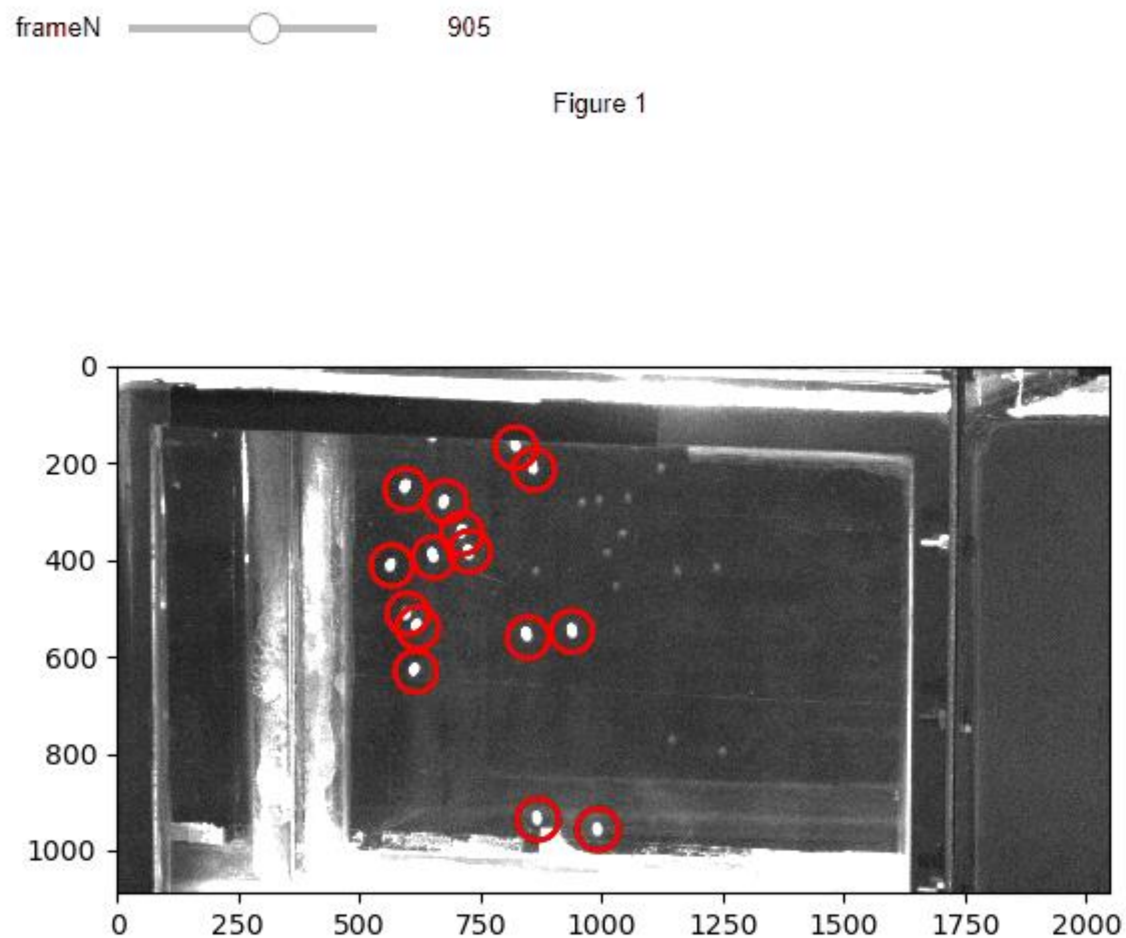
min_mass : 0

Figure 2



User interface design: step 5

- Preprocessing: 2D detection visualization UI.



User interface design: step 5

- Processing: 3D tracking loading interface.

```
PTV.sphere_tracking(871, 896)
```

```
Frame 895: 18 trajectories present.
100%|████████████████████████████████████████| 25/25 [00:15<00:00,  1.61it/s]
```

- Postprocessing : 3D trajectories and velocity visualization.

frameN 872

frameRate 218

Figure 1