Politecnico di Torino
Master Thesis in Computer Engineering
Graduation Session: October 2025

Particle Tracking Acceleration

Supervisor: Riccardo Cantoro
Company Tutor: Kamel Abdelouahab
Candidate: Francesco Risso

Introduction

For the better understanding of what happens during a visual experiment, the possibility of recording the experiment itself and watching it again at a later time is crucial. While for many cases achieving this is just a matter of recording the event with a camera, some complex situations require more attention. One such case is when the nature of the experiments requires to understand 3D data, where a 2D video would not carry the necessary information.

This thesis tackles the task of reconstructing the 3D trajectory of small bubbles moving in the air, observed by a setup of 3 or 4 cameras. The experiment aims to better understand and describe the movement of the air that surrounds us, which emerged as a key factor in understanding the transmission of COVID-19. The expected solution should maximize the reconstruction quality, while always keeping the processing time under control, to have a real-time processing on the captured images.

Solution overview

As commonly done in literature, the final solution splits the task into a 4-step pipeline. Each step is computed in parallel by a separate process, which communicates with the others by means of a shared memory. The captured video traverses therefore the following transformation steps:

- **Locate**: within a single frame, extract the 2D coordinates of the bubbles;
- **3D matching**: combine information of the different cameras to reconstruct the 3D positions;
- **Linking**: join together bubbles from consecutive timestamps into **tracklets**;
- **Visualization**: display the tracklets on a 2D screen, in an intuitive way.

Locate

For the *Locate* step, we used a real experiment to benchmark the multiple approaches available in literature, as well as new algorithms invented within this thesis (the ones marked by * in figure 1). The most important metric was the speed, since the algorithm has to process the frames in real-time. On top of that, quality was evaluated by checking the percentage of bubbles that were missed. Figure 1 compares the different speeds of the approaches with the target speed (3 cameras $\cdot\ 30\ ^{\text{FPS}}/_{\text{camera}} = 90$ FPS): while it may seem that none of them reaches the target, a clean-sheet implementation of the best one managed to reach the target speed. The final choice for the

Locate step is a new algorithm that includes a CPU portion to find the bubbles, and a GPU section to simultaneously find the coordinates of all bubble centroids.

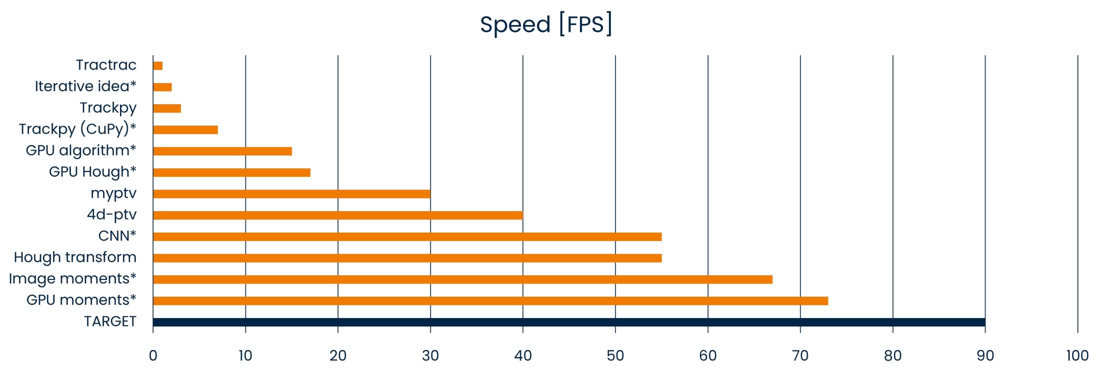

Figure 1: Performance evaluation of different approaches for the *Locate* step.

3D matching

The *3D matching* step is based on the stereoscopy technique, which matches pixels on different cameras to estimate the depth of a 3D point. However, working on plain coordinates instead of textured images added the challenge of correctly matching bubbles across cameras. For this step, no meaningful literature was found: several new algorithms were considered and compared in figure 2. The final choice was the best one among the algorithms whose speed was greater than the required 30 FPS.

Approach	Speed	Correct / Not found / Wrong	
Epilines only	300 FPS	28 / 0 / 2	51 / 4 / 45
Epilines + median (chosen)	55 FPS	27 / 2 / 1	74 / 20 / 6
Epilines + KNN	122 FPS	2 / 28 / 0	1 / 98 / 1
Epilines + short trajectory	125 FPS	12 / 2 / 16	62 / 8 / 30
Epilines + “brute-force” 3D	19 FPS	18 / 12 / 0	55 / 45 / 0
Long trajectory	3 FPS	-	-

Figure 2: Comparing speed and quality of the *3D matching* approaches.

Linking

The *Linking* step could be performed either in 2D, or in 3D. While 2D *link* was more popular in the literature, the results were not satisfactory in terms of accuracy. As a consequence, a 3D *linking* approach was chosen. The final algorithm leverages the fact that the 3D bubbles are quite spaced out, to just link a bubble to the closest one in the next frame, provided that the distance is under a reasonability threshold.

Visualization

Two different options were developed for the *Visualization* step:

- a simple, on-line rendering engine, built using the `Open3D` library. It is able to display the tracklets as soon as they are computed, but it is ephemeral and has little user control;
- a more advanced, off-line rendering engine, developed in the Unity game engine. It offers more control and freedom, but it requires that the reconstruction is complete and stored on disk.

Results

When executed on a NVIDIA Jetson Orin Nano, the pipeline is able to process streams of 4 cameras at 38 FPS. The results are excellent with scenes with up to 100 bubbles within the 3D space under observation. The overall performance fully matches the objectives of this project.

While the speed is already higher than what was required, further optimizations may enable the pipeline to be executed on less powerful and cheaper hardware. In this direction, figure 3 shows that the bottleneck is currently the *Locate* step, which internally is mostly slowed by the loading of the frames to the RAM of the Jetson. Optimizing this memory transfer would therefore enable the overall speed to improve.

Another interesting evolution of this work could be the improvement of the matching quality, to obtain an algorithm capable of analyzing setups with a more dense concentration of bubbles. One of the possibilities to achieve this goal would be to find a better *3D matching* algorithm, or to improve the speed of the *Epilines + "brute-force" 3D* algorithm, which had a better quality, but was too slow. Another direction to improve the *3D matching* would be to have more information on the bubbles: for instance, if the bubbles were of different colors, the *3D matching* step would just need to consider the subset of bubbles with similar color.

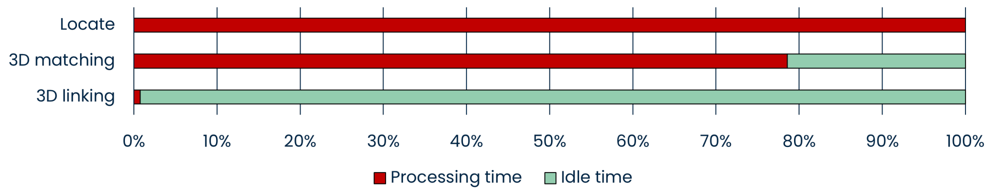

Figure 3: Time spent working and waiting for data, for each step of the pipeline: the bottleneck is the *Locate* step.