April 30, 2013

Abstract

Chapter 1

Introduction

The goal of this thesis is to study and better unstand the dynamics of particles in flows, and is a continuation of two previous theses [**?**][**?**]. The experimental setup is a microfluidic channel where the directional dynamics of microrods are captured using a camera connected to a confocal microscope.

The focus of this thesis is on improving the experimental setup both in terms of the quality of results as well as the efficiency and ease of gathering such results.

The main improvements that were changing particles to a glass material as well as automating the tracking of particles and the flow direction in the channel.

1.1 Background

The dynamics of particles in flow has been studied for many years, with early contributions from Einstein in his 1905 paper "Something something Einstein paper"[**?**], continued by Whatever his name is Jeffrey in 1922 and many since. Jeffrey derived analytically explicit solutions for the motions of triaxial particles in a shear flow called 'Jeffrey Orbits' (see section **??**) which have been used in various theoretical developments. However there is rather limited experimental evidence that such orbits actually exist [**?**]. Previous work by Einarsson[**?**] et al did not satisfactorally show this, presumably because of assymetric particles, possible thermal noise and inaccurate tracking. This thesis has used particles with better symmetry and bla bla bla

1.2 Experimental Setup

In order to study the dynamics of particles, a channel made from PDMS and bonded onto a glass slide. A fluid containing the traxial glass particles that can be seen in fig re REF HERE is pumped at a constant rate through the channel using a INSERT THE PUMP NAME HERE. To observe the particles, the

channel is placed upon a confocal microscope connected to a INSERT CAMERA MODEL HERE. The channel is then moved using a INSERT STEP ENINE NAME HERE controlled from a computer. A mroe detailed description of the setup can be seen in figure REF to SETUP IMAGE THAT I MAKE

1.2.1 The particles

The particles in the experiment are glass and have a width of 5.0+-0.2μm, with lengths distributed as seen in ref THE PICTURE OF THEIR LENGTHS.

1.2.2 Sodium Metatungstate

I guess we use sodium meta tungstate. But does this really need a whole section? Seems like overdoing it... but what

Chapter 2

Theory

2.1 Jeffrey Orbits

2.2 Kalman filter

Chapter 3

Automated Tracking

TODO: READ UP ON THE EFFICIENCY OF THE CANNY EDGE DETECTION TAKE AN IMAGE OF THE STATIC NOISE REDUCTION (DOES IT ACTUALLY WORK?!) GET AN IMAGE OF BEFORE AND AFTER SMOOTHING, PREFERABLY INCLUDE A CANNY EDGE DETECTION OF BOTH ADD ANTONS THESSIS TO BIBLIOGRAPHY WORK MORE ON KALMAN FILTER SECTION WRITE ABOUT THE TIME IT TAKES TO DO VARIOUS STUFFS

3.1 Noise Reduction

The first step in tracking is to correctly identify the particle. To do this we want to eliminate as much noise as we can. The first type of noise we will elimenate is static noise, this is noise present in every image and so results from dirt or scratches on the lens of the camera or microscope. The outside of both would be cleaned fairly regularily, but a significant amount of noise would always remain. To remove this noise, N pictures were taken at different positions in the channel to generate an average image where any particular features of the channel would disappear and only the static noise remain. This can be seen in figure STATIC NOISE REDUCTION IMAGE,

The second form of noise that is eliminated are various thin edges in the image, such as the lines in the PDMS from the scratches made polishing the cast the channel is molded from. Or, possible dirt on the cover glass. To reduce this noise, the image is blurred with a gaussian blur filter of size 7 with $\sigma = 1$. THE RESULTS OF SMOOTHING CAN BE SEEN IN ANOTHER IMAGE

3.2 Canny Edge Detection

The next step is to use the Canny Edge Detection to detect the particles. The Canndy Edge detection is generally considered the most advanced and best performing edge detection of the simple filters. The Canny Edge detection will

first compute a Sobel Filter edge image, by looking at the change in intensity at every pixel from 3 possible directions and averaging these. Then, a pixel is part of an edge if its edge image value is higher than some value, T_{high}. Then all pixels neighboring these edges (and their neighbors) are tested if they are higher than some value T_low. This basically means that you can notice 'weaker' features in an object while at the same time avoiding weak noise in the surrounding area The main drawback of the Canny Edge detection is the computational time, and thus the implementation from the Open Computer Vision (OCV) was used, as this is a heavily optimized routine written in C++ with real time uses in mind. Thanks to this the computing for the Canny Edge of a 260x260 Image takes about X ms and thus is of little importance in the overall time per frame.

3.3 Contour detection and selection

Once an edge image has been generated, the OCV package has another useful function, `Contours`. What this does is basically apply the 'flood fill' algorithm discussed in ANTONS THESIS to every pixel, until no pixel has been covered. We thus get every contiguous group of pixels in the image, which, if we have chosen the threshold values to our edge detection correctly, should hpoefully include that of the particle.

In order to find the correct contour, a few techniques are used to find the correct contour.

First particles whose total size is less than some minimum value, n_{min} or larger than some maximum value $nmax$ are ignored. Then position of each contour $C_i = p_1, p_2 ... p_n$

$$P_i = \sum_j^n p_j / n$$

is compared to the expected position of the Kalman filter, which the very first frame is the middle position. Finally a 'thinness value is computed.'. I am not really sure I should do this now that I have so few particles, but I do it none the less!

3.4 Kalman Filter

When the particle is at constant motion in the channel the equations of motion give us

$$\begin{bmatrix} x_n \\ v_{x,n} \\ y_n \\ v_{y,n} \end{bmatrix} = \begin{bmatrix} x_{n-1} & +v_{x,n-1} \\ v_{x,n-1} & ... \\ y_{n-1} & +v_{y,n-1} \\ v_{y,n-1} & \end{bmatrix} + \begin{bmatrix} 0 \\ c_{xn} \\ 0 \\ c_{yn} \end{bmatrix} y_n v_{yn} \tag{3.1}$$

We can rewrite this in matrix form as

$$\begin{bmatrix} x_n \\ v_{x,n} \\ y_n \\ v_{y,n} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{n-1} \\ v_{x,n-1} \\ y_{n-1} \\ v_{y,n-1} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ c_{x,n-1} \\ 0 \\ c_{y,n-1} \end{bmatrix} \tag{3.2}$$

Now if we want to transform this to the measured data we have to simply multiply by a constant pixel-to-meter constant, which has been meassured to be around $1/940 = 1.06 \cdot 10^{-3}$

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

Which we can easily re-write as a Kalman filter

$$\hat{\mathbf{x}}_n = \mathbf{A}_n \mathbf{x_{n-1}} + \mathbf{B}_n \mathbf{u}_n + \mathbf{w}_n \tag{3.4}$$

$$\hat{\mathbf{P}}_n = \mathbf{F}_n \mathbf{P}_{n-1} \mathbf{F}_n^T + \mathbf{Q}_n \tag{3.5}$$

with

$$\mathbf{x}_n = [x_n v_{x,n} y_n v_{y,n}]^T \tag{3.6}$$

$$\mathbf{A} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

$$\mathbf{B}_n = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

$$\mathbf{u}_n = \begin{bmatrix} 0 \\ c_{x,n-1} \\ 0 \\ c_{y,n-1} \end{bmatrix} \tag{3.9}$$

and we add the update part as

$$\mathbf{K}_n = \hat{\mathbf{P}}_n \mathbf{H}_n^T (\mathbf{H}_n \hat{\mathbf{P}}_n \mathbf{H}_n^T) \tag{3.10}$$

$$\mathbf{x_n} = \hat{\mathbf{x}}_n + \mathbf{K}_n (\mathbf{y} - \mathbf{H}_n \hat{\mathbf{x}}_n) \tag{3.11}$$

$$\mathbf{P} = (\mathbf{I} - \mathbf{K}_n \mathbf{H}_n) \hat{\mathbf{P}}_n \tag{3.12}$$

3.5 Stabilizing the tracking

Once a state estimation has been made, we want to adjust the speed of the step engine to, as best as possible, match tat of the particle. This is done by looking

both the position and velocity of the particle and going through the conditional statements shown in figure CONDITIONAL CORRECTION VECTOR.

The goal is to limit the amount of corrections made, as changing the velocity is rather time intensive as discussed in TIME CONSIDERATIONS, as well as keeping the particle stable. There is also no point in trying to completely eliminate movement

3.6 Time Considerations

A higher FPS will allow the particle detection to be better, improve the position saving and allow the particle tracking to be more stable as well. So maximizing the FPS, ie reducing the computational time of each task, is a clear goal for a good automated tracking. A list of the different tasks and their average execution times can be seen in REFERENCE A TABLE OF THAT STUFF HERE

We can clearly see that the limitors of our FPS are two routines: The screen capture routine and the change velocity routine. The first is unavoidable and must be done every frame by definition if we are interested in knowing the particles position. The second we can use more sparsly, but since the time constraint is in the communication with the step engine, there is not any optimization to be done here either.

Chapter 4

Experimental Setup

Chapter 5

Results

Chapter 6

Discussion