May 6, 2013

Abstract

Chapter 1

Introduction

The goal of this thesis is to study and better understand the dynamics of particles in flows, and is a continuation of two previous masters theses [?][?]. The experimental setup is a microfluidic channel where the rotational 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 have been to replace the particles 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

Previously the particles used in the experiment were polymeric made by swirling an apoxy solution in a vortex. This produces rod like particles of different lengths, but very often included noticeable defects such as being bent, small lumps etc. To improve results glass particles from Nippon Glass company in Japan were brought in. As they are mainly manufactured to serve as spacing rods in LCD screens they have a very well determined width but then broken at varying lengths. These breaking points are the only real visible point of imperfection of the rods. The specification of length and width can be seen in figure **??**

While the glass rods are far more regular in terms of size and shape, they do have on major problem. The density of the glass is 2.57 g/cm^3, significantly higher than water which has a density of around ENSITY OF WATER at 20 degrees C and glycerol which has a density of DENSITY OF GLYCEROL. And so the particles will sink very quickly in any mix of water and glycerol.

The solution is to use Sodium Metatungstate, water soluble mineral which gives the resulting mix a maximum density of around DENSITY OF SOMT which is more than sufficient to keep the particles suspended. It is however not without its own issues which will be discussed in the discussion portion of the thesis.

First the viscosity of the mix is significantly lower than the glycerol and water mix used previously (a 1:4 mix of glycerol and water was commonly used) which means the system is more sensitive to noise. Second the density of the solution changes when the water mixed in evaporates. As the ratio of water to mineral is relatively high, clearly more than 1:1 by weight, a small percentage of water evaporating will result in significant change in density

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 a particle is to correctly identify it in the image given. To do this we want to eliminate as much noise from the image as we can. The first type of noise we will eliminate is static noise, this is noise present in every image and results from dirt or scratches on the lens of the camera or microscope. The easiest solution to such noise would of course be to clean the instruments, but despite numerous attempts of carefully cleaning every surface a noticeable of static noise would always remain.

This means one must use algorithmic noise reduction methods to remove this noise. The method employed is a simple averaging that takes N pictures 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 eliminated are objects of no interst, such as the lines in the PDMS channel, 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 Canny Edge invented by John Canny in his 2011 paper the canny of cane [**?**]detection is generally considered the most advanced and best performing edge detection of the simple filter functions. Without going in to the finer details, given an image matrix **I** where each value corresponds to the light intensity $i_{x,y}$of that pixel at index x, y the Canny edge detector will try to find the cohesive pixels $E = \{e_1, e_2...e_n\}$ where there is a noticeable change in intensity. In other words, what we call an "edge" an image.

This is accomplished in two steps. First a Sobel Filter edge image is computed by looking at the change in intensity at every pixel from 3 possible directions and averaging these.

S = MATHEMATICS OF SOBEL

We then consider a Then, a pixel

$p_{x,y} \in E \text{if} S(x, y) > T_{high}$

where T_{high} is a predetermined threshold value.

Secondly we recursively check all pixels neighbouring an edge

MATH?

if they are higher than some threshold value $T_low.$ if they are, they are also considered part of the edge. This is repeated until no more pixels are added. An image illustrating this process can be seen in figure FIG. The benefit of the Canny edge detection over the simpler Sobel is that it is much easier to detect cohesive objects that vary in intensity without getting a lot of other noise in the iamge. The only real drawback of the Canny Edge detection is the computation 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.

After noise reduction the Canny Edge Detection is used to find the most significant edges in the image.

3.3 Contour detection and selection

Once an edge image has been generated, the OCV package has another useful function, `Contours` which returns a list of every contiguous group of edge pixels. If we have chosen the threshold values to the edge detection correctly, this should include the particle or a good approximation of it.

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 n_{max} are ignored. Then the position pP_i of each contour $C_i = p_1, p_2...p_n$ is calculated as the average pixel position

$$P_i = \sum_{j}^{n} p_j/n$$

This position is compared to the expected position of the Kalman filter, which the very first frame is the middle position.

Finally a 'thinness value' is calculated according to eq 3.1

$$w_{thin}\left(\frac{n}{d_{max}^2}\right)^2 \tag{3.1}$$

. where w_{thin} is a weighting constant, n is the number of pixels in the contour and d_{max} is the longest distance between two pixels in the where 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

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

We can rewrite this in matrix form as

$$\begin{bmatrix} x_n \\ v_{x,n} \\ y_n \\ v_{y,n} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{n-1} \\ v_{x,n-1} \\ y_{n-1} \\ v_{y,n-1} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{matrix} 0 \\ c_{x,n-1} \\ 0 \\ c_{y,n-1} \end{matrix} \tag{3.3}$$

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 $1/940 = 1.06 \cdot 10^{-3}$

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

Which we can easily re-write as a Kalman filter

$$\hat{\mathbf{x}}_n = \mathbf{A}_n \mathbf{x}_{\mathbf{n-1}} + \mathbf{B}_n \mathbf{u}_n + \mathbf{w}_n \tag{3.5}$$

$$\hat{\mathbf{P}}_n = \mathbf{F}_n \mathbf{P}_{n-1} \mathbf{F}_n^T + \mathbf{Q}_n \tag{3.6}$$

with

$$\mathbf{x}_n = [x_n v_{x,n} y_n v_{y,n}]^T \tag{3.7}$$

$$\mathbf{A} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

$$\mathbf{B}_n = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

$$\mathbf{u}_n = \begin{bmatrix} 0 \\ c_{x,n-1} \\ 0 \\ c_{y,n-1} \end{bmatrix} \tag{3.10}$$

and we add the update part as

$$\mathbf{K}_n = \hat{\mathbf{P}}_n \mathbf{H}_n^T (\mathbf{H}_n \hat{\mathbf{P}}_n \mathbf{H}_n^T) \tag{3.11}$$

$$\mathbf{x_n} = \hat{\mathbf{x}}_n + \mathbf{K}_n(\mathbf{y} - \mathbf{H}_n \hat{\mathbf{x}}_n) \tag{3.12}$$

$$\mathbf{P} = (\mathbf{I} - \mathbf{K}_n \mathbf{H}_n)\hat{\mathbf{P}}_n \tag{3.13}$$

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 that 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 table 3.6

Task	Average time	Std deviation
Capture screen	1000	200
Find edges	200	20

We can clearly see that FPS is limited primarily by three routines: The screen capture routine, the change velocity routine and finally the save position routine. The first and last are unavoidable and must be done every frame by definition if we are interested in knowing the particles position as well as possible. This means we simply want to use the velocity correction as little as possible. Since the time constraint is in the communication with the step engine, there is not any optimization to be done here, at least not within the scope is this thesis.

Chapter 4

Experimental Setup

Chapter 5

Results

Chapter 6

Discussion