Exploring the uses of the SIFT algorithm

Marc Parcerisa, Anna Planella

***Abstract*—This article presents a video tracking method with possible future applications to measure a fluid's viscosity by tracking its front while moving through a micro-scale canal. A microscope video of the rheometer's canal with the fluid front advancing through it is analyzed through computer vision and image processing. A tracking method is proposed, based on the SIFT algorithm, to detect interest points on each frame and proceed with a feature matching technique to track them throughout the video. Once this is done, by extracting the tracker speed data and the geometry of the canal, a precise measure for the static viscosity could be determined. The algorithm discussed and experimented with here doesn't allow for such final calculations, but it still gives insight to these methods and presents the idea for a future success.**

***Index Terms*—Interest points; Invariant features; Object recognition; Scale invariant; Image matching; SIFT; K Nearest Neighbours; Viscosity; Navier-Stokes**

I. Introduction

THE task of selecting and tracking objects of interest throughout a large amount of images, or frames, is no short of a difficult task. There are many different algorithms and methods that have been proposed and polished for many years in an attempt to achieve a robust and universal form of video tracking, with varying results, [3] [5] [6].

In this article, however, we attempted to focus on a single application of such techniques to create an algorithm that gave us the desired precision on the specific problem we wanted to tackle. We will combine SIFT [7], and a *K Nearest Neighbours* algorithm to match SIFT descriptors to record a fluid front moving through a micro-scale canal with the hopes of implementing it in the future on an automatic micro-viscometer for blood.

This paper is organized as follows. In Section 2, we present the motivations to this program, discussing briefly the physics of the micro-viscometer and why is it possible to determine a fluid viscosity by tracking it on a microscope video. Section 3 roughly describes the workings of the SIFT algorithm and our use of it. We explain on Section 4 how the *KNN* algorithm works to make the matches between SIFT features from the reference image (selection on a still frame) and features from each frames. Then, on Section 5 we discuss three experiments we made on three videos with extremely different conditions to put the algorithm to the test. It is on this Section we find out the program proposed isn't really suited for the scope of this paper, but does work correctly on other videos. Section 6 gives a conclusion for this article.

II. Motivation: Fluid front micro-rheology

This paper is inspired by the work of Claudia Trejo and her team on moving front microrheology [4]. To make this paper more self-contained we will now explain its principles:

Suppose a rectangular canal of width w and height b. On one end, a nylon micro-tube is connected from where the fluid will enter. This fluid is provided by a large fluid pool filled at a known height that is connected to the other side of the nylon tube. The setup can be seen in figure 1, where H is the height from the micro-canal to the fluid-air interface on the pool, L_T is the tube's length, R is its inner radius, and w and b are the canal's width and height respectively, as previously mentioned. It also shows the position of the camera recording the fluid from a great enough distance to reduce any affinity on the edges. By solving the Navier-Stokes equations both in

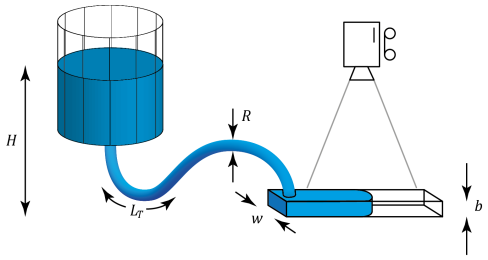


Fig. 1. Experimental setup for a moving front micro-rheology viscosity measurement.

the tube and in the canal, we can calculate the fluid speed of both:

$$U_T(r) = \frac{1}{4\eta}\left(\frac{\Delta P_T}{L_T} - \rho g\right)\left(R^2 - r^2\right) \tag{1}$$

$$U_C(z) = \frac{1}{2\eta}\frac{\Delta P_C}{h}\left(bz - z^2\right) \tag{2}$$

Where η is the fluid viscosity, ΔP is the increase of pressure along the tube (T) or canal (C), ρ is the fluid density, g is the gravitational acceleration, r and z are the radial and vertical coordinates on the tube and canal respectively, and lastly, h is the distance travelled on the canal by the fluid.

By considering the average fluid velocity over both duct sections, the following expressions are calculated:

$$\langle U_T \rangle = \frac{R^2}{8\eta}\left(\frac{\Delta P_T}{L_T} - \rho g\right) \tag{3}$$

$$\langle U_C \rangle = \frac{b^2}{12\eta}\frac{\Delta P_C}{h} \tag{4}$$

By solving for both pressure increases, considering all isobar surfaces on the system, and given that $\langle U_C \rangle \cdot bw = \langle U_T \rangle \pi R^2$, a relation between the fluid viscosity and the mean canal velocity can be found:

$$\eta = \frac{\rho g H - \Delta P_{cap}}{\frac{8bwL_T}{\pi R^4} + \frac{12h}{b^2}} \cdot \frac{1}{\langle U_C \rangle} \tag{5}$$

Where ΔP_{cap} is the capillary pressure on the fluid-air interface, which can be determined by looking at the angle of contact at the canal walls. Finally, considering that $bwL_T \gg R^4$, the above expression can be approximated as:

$$\eta = (\rho g H - \Delta P_{cap}) \frac{\pi R^4}{8bwL_T} \frac{1}{\langle U_C \rangle} \tag{6}$$

III. The SIFT algorithm

SIFT (Scale-Invariant Feature Transform) [2] is a method proposed to extract distinctive features from our images, to later match them between the different frames. It is presented as robust, reliable and invariant to many changes such as rotation, illumination or noise.

This algorithm already has its own readily available functions, which we will use to obtain our points of interest so we can later extend on a proper matching and tracking technique.

A. A small explanation

What follows is a small and superficial explanation of how the SIFT works, before moving on to the bulk of the project. There are four steps involved in the algorithm:

1) Scale space extreme detection
2) Key-point localization
3) Orientation assignment
4) Key-point descriptor

The first step is key when tracking objects that may vary in scale or size. Since we are limiting the applications of our method, this shouldn't be a problem to be concerned with. However, to give a brief explanation, the Difference of Gaussian (an approximation of *Laplacian of Gaussian*) is found for the image with different values of σ, which detects blobs of different sizes. This way it finds the local maxima across the scale and space.

Once potential key-points are found, a Taylor series expansion of scale space is used to get an accurate location, and if the intensity at this extreme is less than a threshold value it is rejected. The prior approximation has a higher response for edges, so edges also need to be removed. This eliminates any low-contrast key-points and edge key-points and what remains is strong interest points.

Orientation assignment is done by taking a neighbourhood around the points to calculate the gradient magnitude and direction is calculated in that region. The group of most frequent directions is taken to decide the final orientation.

Finally, the descriptor is created taking, once again, a neighbourhood of the point, dividing it into "bins", and assigning a direction to each. These create a vector.

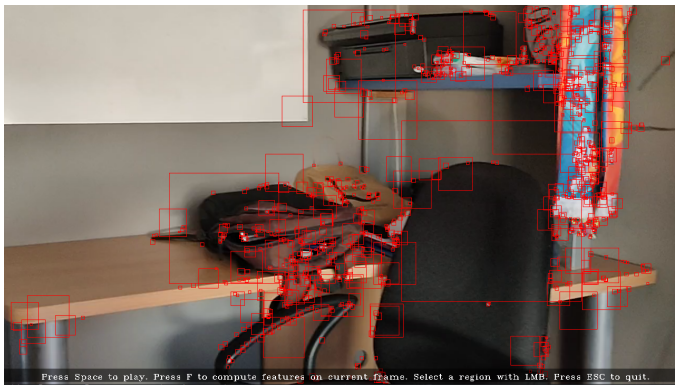


Fig. 2. What the SIFT algorithm ends up identifying as *features* in a random frame of a video.

B. The SIFT class

As previously mentioned, our program does not include the entirety of the SIFT algorithm in a hard-coded way, since that would be detrimental to the overall efficiency and quality of the program. There is a class in *OpenCV* which extracts key-points and computes the descriptors according to this algorithm - it is this one we will be using.

IV. Feature Matching using KNN

The *K Nearest Neighbours* algorithm takes the descriptor of one feature in a set and it compares its euclidean distance to all the others in a second set. It returns the k ones that are closest [1].

So key-points between two images are matched by identifying their nearest neighbours on the descriptor space. That brings us to the problem that, in some cases, the second closest-match may be very near to the first and the amount of noise in the picture can have a lot of impact in such situations. In that case, ratio of closest-distance to second-closest distance is taken. If it is greater than a threshold (0.8 is the usual take), they are rejected. This eliminates around 90% of false matches while discards only 5% correct matches, as per the paper [2].

The *BFMatcher.knnMatch()* function gets the k best matches to each descriptor. We can take $k = 2$ to be able to apply the aforementioned ratio test. We will make the cut at 0.7. Once we have a group of good features for each frame, we will compute the average position and the corresponding error to draw an on-screen marker.

(a) (b) (c)

Fig. 3. (a) Frame paused for selection along with a frame where it went undetected. (b) A frame where it detected the ball in the dark. (c) A frame where it detected the ball in the light.

(a) (b) (c)

Fig. 4. (a) Frame paused for selection along with the same frame catching the feature. (b) A frame where there error bars got bigger. (c) A frame where it still detects the object, even when it is almost out of frame.

V. Tests

Our program will allow us to play and pause any video of our choice through an interface to find our frame of interest. It will then allow us to open a separate window were the object or point of interest can be selected. The program will tell us if there are any *features* in that space, which is what it is really looking for, and how many. If we play the video after making such a selection, the average position of the matching features inside that zone will show in every frame, hopefully tracking our target.

We have tested this program with various videos.

A. A video of a ball

We started by testing our program with a video of a contrasting-colored object in a plain background. The tracking was decently stable when the object was not moving too fast and lit in the same way as the selection frame, see Figure 3c. It detected the object under a different light but the tracking wasn't fluid during that part of the video (3b). It also didn't detect any features on seemingly normal frames (3a).

B. Someone's room

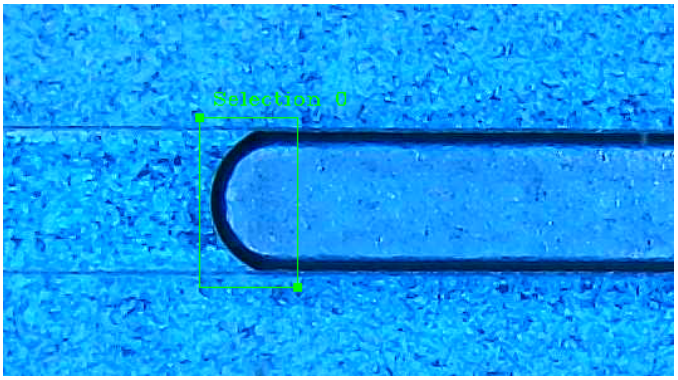
What has probably been our most successful test has been this one of someone's desk. Even though no colours stand out our program was able to follow most of the objects, regions or edges we selected, see Figure 4a. Furthermore, when the object was out of frame, it hardly ever "matched" our selection to some far gone feature and stated instead that "no matches were found". The error bars of the objects kept varying in sizes during the video but not to the point were it was unreliable and the main point stayed reasonably stable (4c).

C. Fluid front

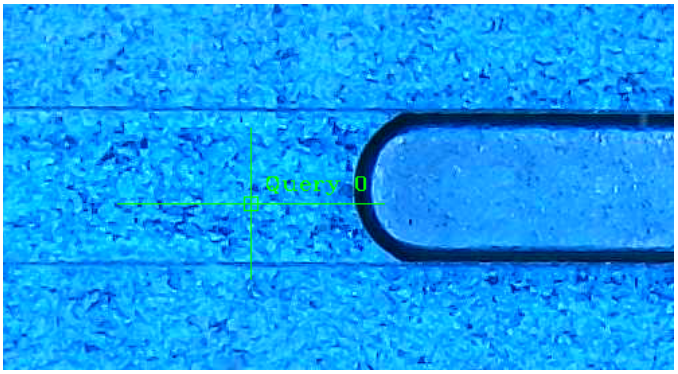
Finally, the experiment this program was designed for. When trying to track the moving front on the video, the program lost it very rapidly and acted as if the target wasn't moving.

Upon close inspection of the features it was following, we hypothesized the cause of the problem: the microscope video of the front has a very dense granulation, which is given by the roughness of the transparent chip's bottom layer (Figure 5). This is caused by the 3D printer polymerization process, which is a part of the process we have no control over. This noise on the image is interpreted as a lot of small feature points by the SIFT algorithm, causing the program to start tracking them instead of the few interest points given by the fluid front itself. That is why, when trying to track a big enough window to contain some features, the program will respond as if it is pretty much stationary.

We have tried to fix this issue by adding layers of pre-processing to this video: we cut it to minimize the grainy



(a)



(b)

Fig. 5. (a) Selection on a video of a fluid front on a micro-rheometer. (b) Tracked position of the fluid front stays in exactly the same position as it was when the selection was made.

surface and focus it only on the canal and we also added some Gaussian blur to lower the noise. This last step proved to be an improvement. The tracker now didn't stay at the start of the frame. However, even though it tracked the fluid walls, it barely stayed on the same region. Since our goal was to measure the speed of the front, a jumping tracker was as useless as none.

VI. Conclusion

The original motivation of this article was to be able to measure a fluid's viscosity through video tracking. As it has been exposed in the latter section, some variables out of our control have made it very difficult to achieve that, and the method proposed isn't the best suited. It is possible that some other algorithms, probably more related to contrast or color, have a much higher efficiency with the original problem.

Our method by itself, however, did great in other areas. Due to the matching method, it is not accurate with objects with high speed. However, it is able to keep track of more static, even if more subtle, objects and details.

References

[1] Oliver Kramer. *K-Nearest Neighbors*. 2013. DOI: 10.1007/978-3-642-38652-7_2.

[2] David G. Lowe. "Distinctive image features from scale-invariant keypoints". In: *International Journal of Computer Vision* 60 (2 Nov. 2004), pp. 91–110. ISSN: 09205691. DOI: 10.1023/B:VISI.0000029664.99615.94.

[3] *Mean-shift Blob Tracking through Scale Space*. URL: https://ieeexplore-ieee-org.sire.ub.edu/stamp/stamp.jsp?tp=&arnumber=1211475.

[4] C Trejo-Soto et al. "Capillary Filling at the Microscale: Control of Fluid Front Using Geometry". In: (2016). DOI: 10.1371/journal.pone.0153559.

[5] CJ Veenman, Ea Hendriks, and MJT Reinders. "A Fast and Robust Point Tracking Algorithm". In: ().

[6] *Wide-range Feature Point Tracking with Corresponding Point Search and Accurate Feature Point Tracking with Mean-Shift*. URL: https://ieeexplore-ieee-org.sire.ub.edu/stamp/stamp.jsp?tp=&arnumber=6778462.

[7] Huiyu Zhou, Yuan Yuan, and Chunmei Shi. "Object tracking using SIFT features and mean shift". In: (2008). DOI: 10.1016/j.cviu.2008.08.006.