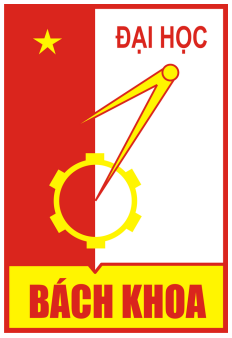

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

IT3930E - 718444 - Project II

Data Generation for Deep Spray

Faculty advisor

PhD. Dinh Viet Sang

Student

Tran Quoc Lap - 20194443

HANOI 2022

Contents

Chapter 1

Introduction

The final goal of this project is to, given a simulation of liquid flow, detect and analyze different droplets. There are 5 classes that need to detect detached: bag, lobe, ligament, drop and attached ligament. This problem belongs to the class of object detection problems. Currently, there are 2 possible approaches.

The first direction is to use traditional image processing techniques. In this direction, we will set up a set of rules based on visual observations. The advantage of this approach is that, there is no need for labeled training data. But the downside is, because it is based on only a finite set of rules, it may only work for simple situations.

The second direction is to use deep learning. In this direction, we build a neural network and use a labeled dataset for the network to learn on its own. The advantage of this method is that the network can learn the hidden knowledge and build a complex set of rules far beyond human comprehension. Therefore, it is expected to be able to handle complex cases. But the downside of this approach is that it requires a lot of labeled data.

The problem is that, after some trial and error, it is impossible to detect all the objects purely by traditional image processing techniques. It can only handle simple cases like isolated drop, but is powerless in more complex cases like ligament or overlap. I believe those difficult cases can only be solved by deep learning. Since the current image set is completely unlabeled, my goal in this project is to generate training data for deep learning. Specifically, I will use traditional image processing techniques to detect and label some objects, and then use them to generate training data for deep learning.

However, this approach means that the neural network will be trained on a completely dummy data set, but will infer on the real data. This requires the generated data to be very similar to the real data.

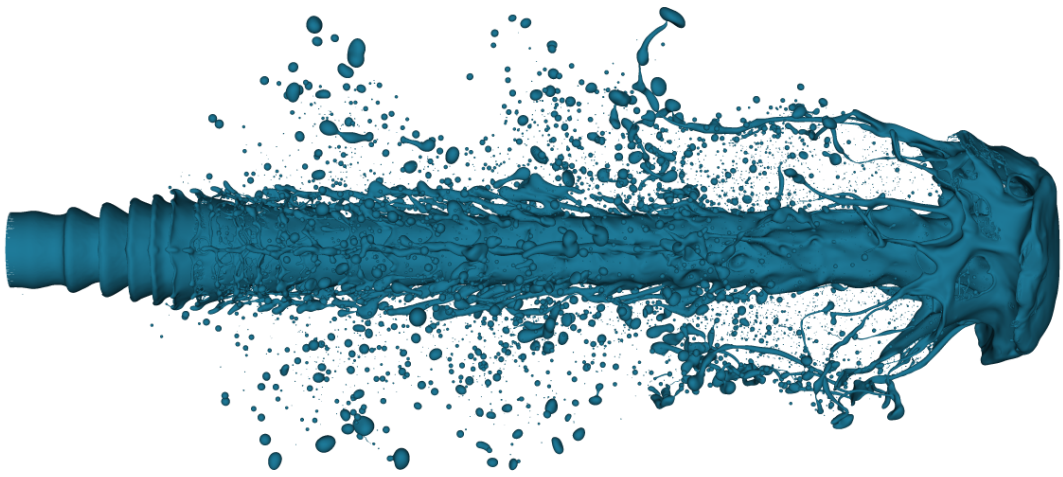

Figure 1.1: Sample image of liquid droplets.

Chapter 2

Method: Strategy 1

The strategy I outlined is described in Figure 2.1.

My intention is, from the input image without labels, I will filter the objects that are separate from the main flow (task (1)). The main flow and the objects lying on top of it (including drops, detached ligaments, and attached ligaments) is a hard cases. In task (2), I will collect isolated drops using some simple rules, the rest are detached ligaments and cluster of conjoined objects. In task (3), I will filter the detached ligaments using another set of rules. The rest that remains unresolved after task (3) is the cluster of conjoined drops, and I will try to separate them in task (6). At this point, I will be able to synthesize attached ligaments from detached ligaments, and then paste them into the main flow to generate training data for deep learning.

However, in reality, I am stopped at task (2), because the image processing techniques I use are not capable of further processing. Section 2.1 through **??** will explain in more details.

2.1 Task (1): Select objects separated from main flow

Task 1: Filter for objects that have no connection to the main flow. This is simple because we can use the contour finding algorithm to filter obtain them. Contour finding algorithm really works, because the liquid flow is sprayed on an all-white background with no noise, and the color of the liquid is completely separable from the background color. However, the current contour finding algorithm cannot work on color or black and white images. It only works on binary images. Therefore we need to either extract edges from the original image, or binarize the image before finding contours.

In the first approach, I used Canny for edge extraction but the results were not very good. Two problems arise. Firstly, Canny is not suitable for catching tiny drops. With droplets that are only a few pixels in size, Canny can miss them. For the slightly larger drops that Canny can catch, the boundaries of these drops were

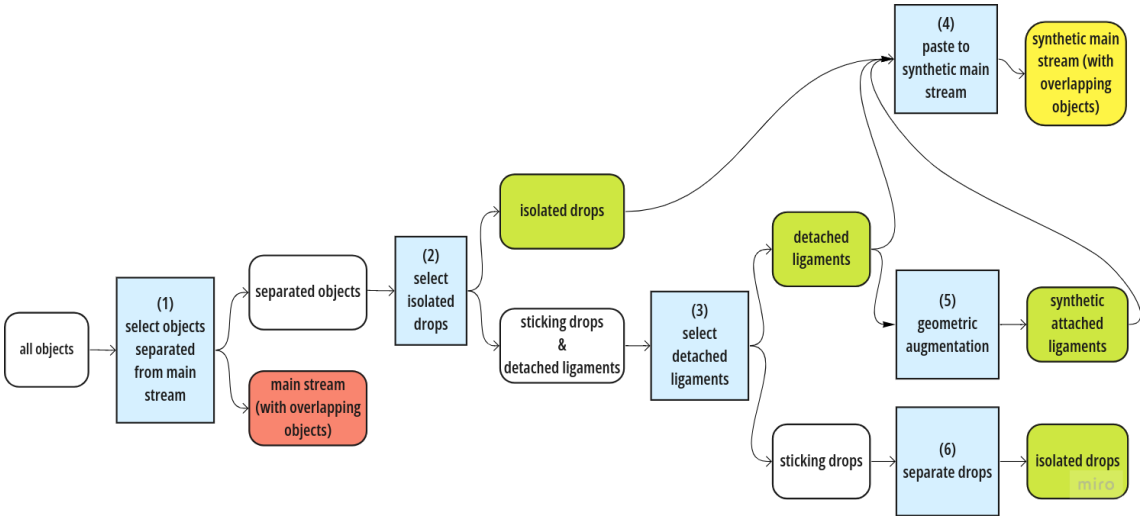

Figure 2.1: The diagram of strategy for data generation.
The blue rectangles describe the set of rules we define to filter objects. The green rectangles are objects that have been filtered and labeled by those set of rules. The red rectangles are hard cases, they are not likely to be solved by our finite set of rules, thereby will be solved by deep learning. The yellow rectangle depicts the synthetic image that we will feed as training data for deep learning to solve the hard cases in the red rectangles. They are synthesized using objects that we have successfully filtered and labeled.

expanded, making the drops in dense areas get conjoined to each other. Secondly, we expect the edge features to be very smooth and circularly closed, but in my experiments I found that quite a few of the edges generated by Canny are open (see Figure 2.2). This causes the contour to turn back at the verge of the exposed, eventually the contour is very different from the real shape of the objects.

The second approach, which generates a binary segmentation of image (Figure 2.3), is more efficient. First, converting to binary is easy because I only need to choose a global threshold to segment the fluid from the white background. Second, all the problems that Canny encountered such as not catching the tiny drops, expanding the boundary of the drop or opening the border were not encountered.

Therefore, I decided to detect contours on the binary segmentation image to filter for the drops that are separate from the main flow.

2.2 Task (2): Select isolated drops

Among the objects that are filtered out in step 1, there are drops that are alone and do not stick, do not overlap with any other objects. Since the geometry of drop is the simplest, we will set up a set of rules to filter them.

Initially, I used the law of aspect ratio: objects with bounding boxes that have an aspect ratio less than 3 are considered drops. However, this rule produces many

Figure 2.2: Case where Canny edge detector produce open boundary.

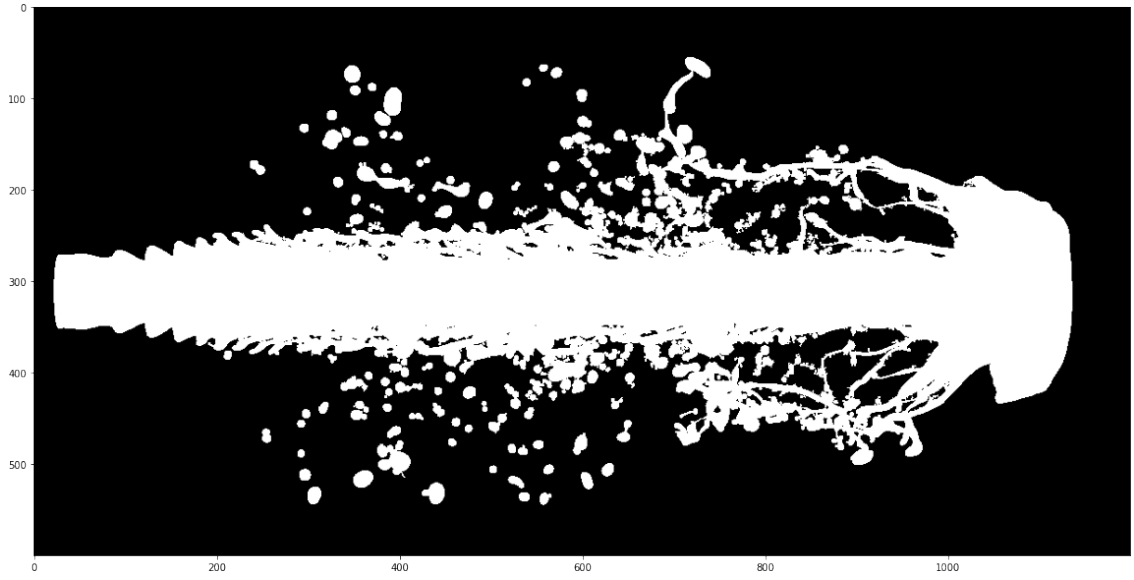

Figure 2.3: Binary segmentation of original image.

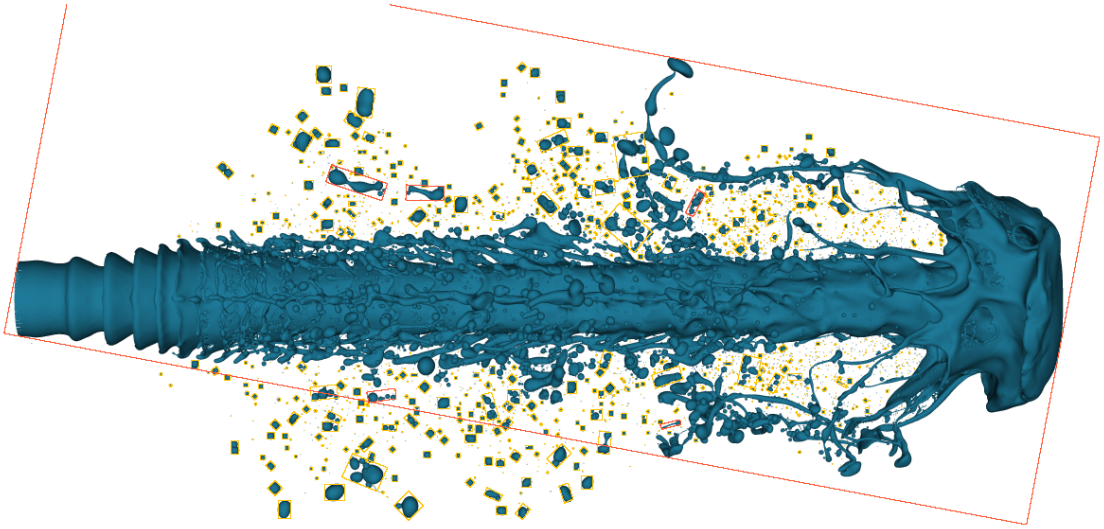

Figure 2.4: Result after applying aspect ratio criteria. Note that very little case are marked red (non drop), many conjoined object are marked as a drop.

false positives, because some cluster of conjoined objects also have an aspect ratio less than 3, sometimes even close to square (Figure 2.4). I find that using the aspect ratio rule covers very little of the geometric characteristics of the object. An object if has a aspect ratio ratio less than 3, but in order to be a drop it needs to look round and full. So I experimented with Hu moment.

Hu moment can be used to find the distance between two shapes. If the distance is small, the shapes are close in appearance and if the distance is large, the shapes are farther apart in appearance. So in order to use Hu moment in filtering the drops, I manually selected the most typical drops as a template, then measured the Hu moment distance between the objects in step 1 with these 2 templates and selected an appropriate threshold do decide whether an object is a drop. Since the ligaments are often long and curvy, and a cluster of conjoined objects often has odd shapes that aren't as round and firm like a drop, this approach has proven to be quite effective (Figure 2.6). After some testing, the number of templates I chose was 2 (Figure 2.5).

2.3 Task (3): Select detached ligament

In this step, I will filter the ligaments from the unprocessed objects in task (2). In other words, in task (3) I need to distinguish ligaments from clusters of conjoined objects.

I have tried many methods but none of them actually works. At first, I tried using shape matching again with more templates, but the shape of both the ligament and the conjoined objects is so diverse that there are too many false positives and false negatives.

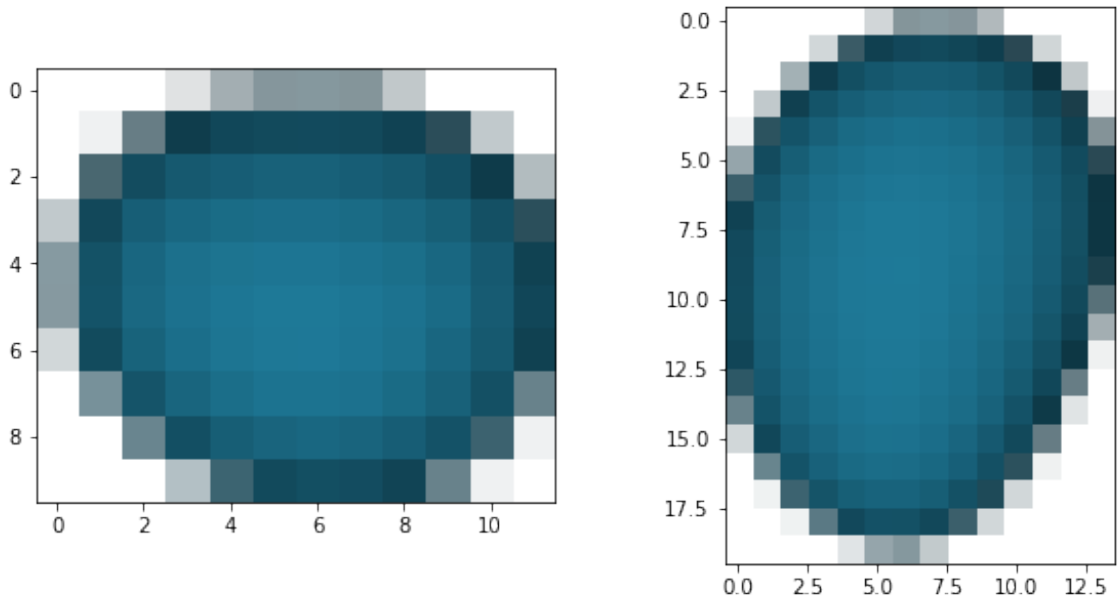

Figure 2.5: Template for shape matching.

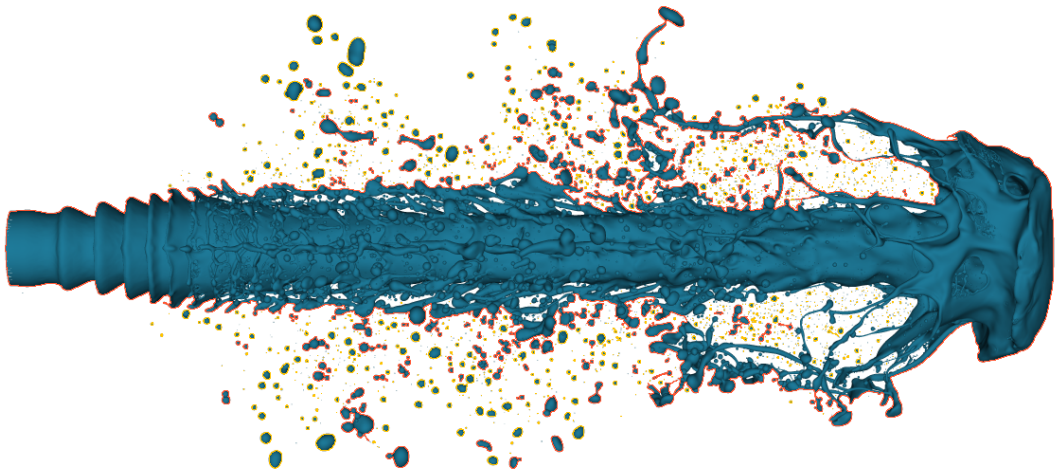

Figure 2.6: Result after applying shape matching criteria.

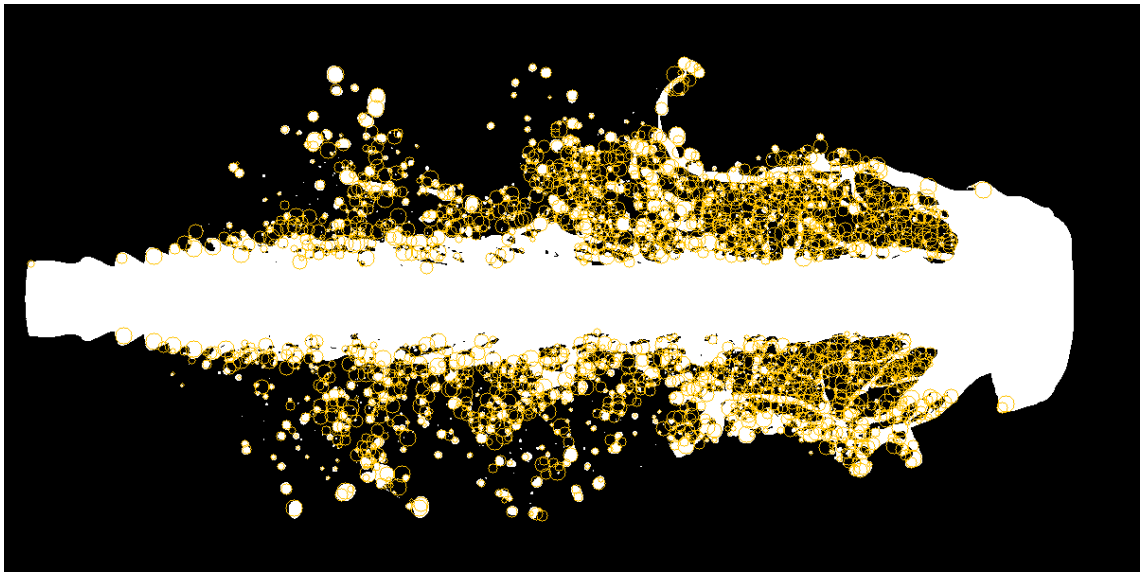

Figure 2.7: Result after applying Hough circle transform.

Another approach that I tested is based on the law of elasticity and smoothness of contour. At first I believed that ligaments, regardless of their shape, should have a certain smoothness and elasticity in their borders, rather than kinking like those created by conjoined objects. The elasticity and smoothness here are different from convexity. An elastic and smooth contour can be convex or concave, it simply doesn't have sudden twist in the edge direction. However, I can't find any function in OpenCV library that can calculate this elasticity and smoothness. I tried trying to develop one by myself but it didn't work.

The two ineffective approaches above have one thing in common: discerning a special feature of the ligament over other objects. That made me come up with another idea, which is to move on to finding the special feature of conjoined objects. In details, we know about the Opening operation - it is used on binary images to remove small objects such as noise or unwanted holes. I believe in a cluster of conjoined objects, there are some small drops, some larger ones. If we use the opening operator just enough to dissolve the small drops, the size of the large drops after opening won't change much. And if we take the original mask of the clustered objects and subtract its mask after opening, the remainder will be of the size with the smaller object. Therefore, I try to apply this method to all objects in task (2), in case there is a small residual, it is a ligament, in case there is a large residual, it is a cluster of conjoined objects. Again, however, this doesn't work, because the sizes of the drops in the cluster of conjoined objects can be so close that they both dissolve, and the sizes of the objects are so varied, that for can't find some generic number of iteration for the open operation.

As the last attempt, I tried the Hough circle transform to separate the conjoined drops. However, this only handles very few cases and often produces very noisy results (Figure 2.7).

After all, I was not able to solve task (3) using traditional image processing techniques. Up to this point, I can only filter out isolated drops by globally thresholding and shape matching using Hu moment. So, I switched to another plan, see section 3.

Chapter 3

Method: Strategy 2

In this strategy, I take a more modest approach. In the previous strategy using traditional image processing techniques, I was stuck at filtering for the ligaments. In the second strategy, I will let the neural network learn to detect them, by synthesizing manually labeled ligaments. For details see Figure 3.1.

3.1 Task (3): Select isolated drops

The goal of this task is the same as of task (2) in the 2.2 section. However, I would like to try a different approach. Instead of picking out templates and comparing Hu moment distances, I wanted to set some more specific criteria for drops. Specifically, I first filter out all objects whose area is less than a specific threshold, which ensures that it is a drop. Then I require the remaining drop to satisfy a minimum circularity and convexity level. Since the drops are usually quite round and forward convex, I believe these 2 rules will probably work well. And it actually works pretty well (see Figure 3.2). With the naked eye, I don't discern any difference in accuracy between using these two criteria and using template shape matching as in the first strategy. I suspect the choice of which way is down to preference.

3.2 Task (4): Geometric augmentation for labeled detached ligaments

After manually labeling detached ligaments in task (2), beside combining them with labeled isolated drop in task (3) to synthesize conjoined objects, I use some augmentation to synthesize attached ligaments.

The first step is to resize the detached ligaments so that they are roughly the same size as the attached ligaments. I chose a random $\frac{\text{length}}{\text{width}}$ ranges between $(5, 11)$, and magnified it to no more than the size of the largest attached ligament I observed. Then, to create synthetic attachec ligaments that are more varied in shape, I apply affine transform piecewisely (see Figure 3.3).

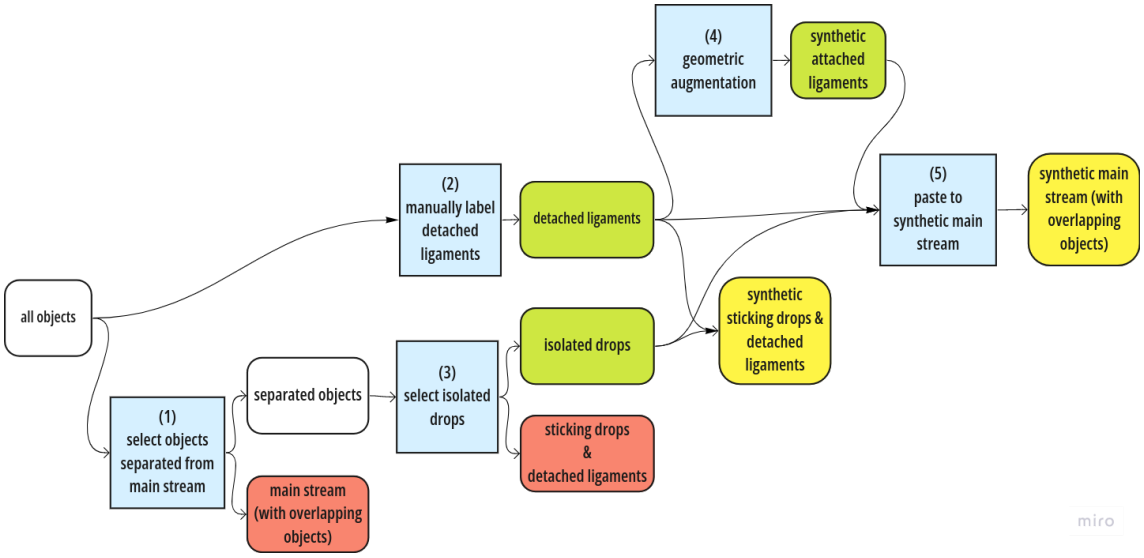

Figure 3.1: The diagram of the second strategy for data generation

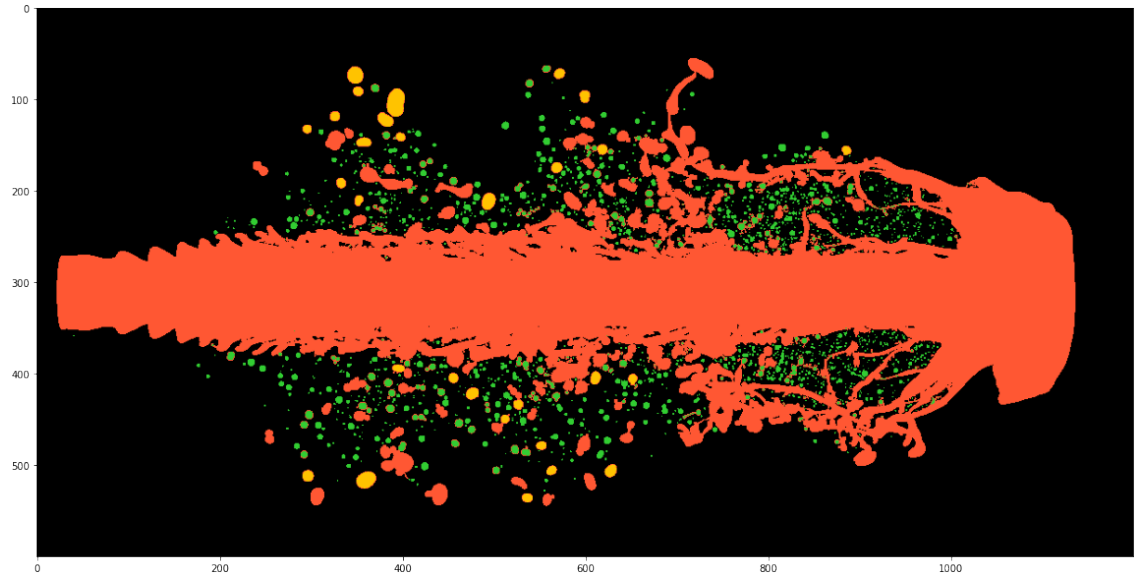

Figure 3.2: Visualization of filtered drops using blob detection. Green color denotes drops filtered using area criteria. Yellow color denotes drops filtered using circularity and convexity criteria. Red color denotes unprocessed cases.

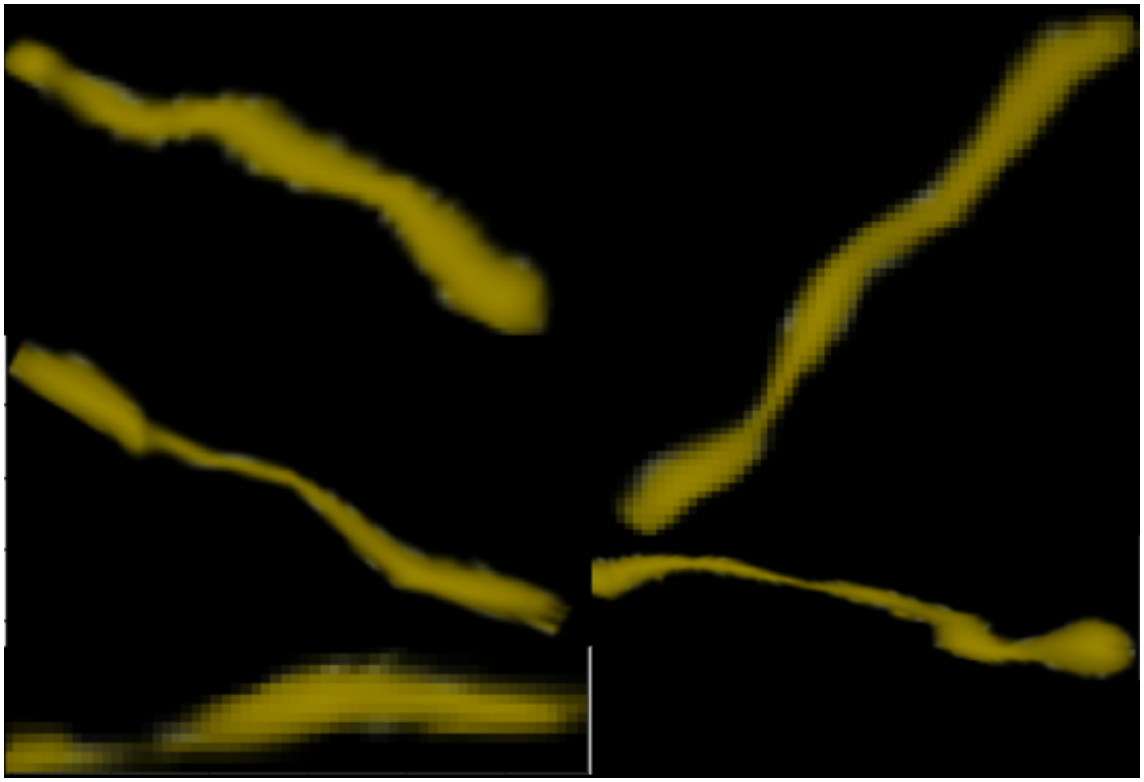

Figure 3.3: Detached ligaments after piecewise affine transform.

3.3 Task (5): Paste labeled object to a synthetic main flow

This is the final step in generating the training data. Up to now, we have

- isolated drop labeled using algorithms in task (3)
- detached ligaments manually labeled from task (2)
- and synthesized attached ligament from task (4)

Before we get to work, we need a main flow - the one that doesn't have any other objects on top of it so we can paste the labeled objects we already have. To do that, I removed all separated objects from the original image in task (1), then applied a median filter to remove all unwanted objects (see Figure 3.4). At this point, we can start pasting.

Pasting drops and detached ligaments is very easy. I can paste it almost anywhere in the image without any further modification. The difficulty lies in the attached ligaments. First, most of them must be attached to the primary breakup of the flow, and one end of each must be attached to the main body of the flow. Second, the pasting procedure should not produce a visible seam, we need a seamless pasting in order to distinguish it from detached ligaments which overlap the main flow.

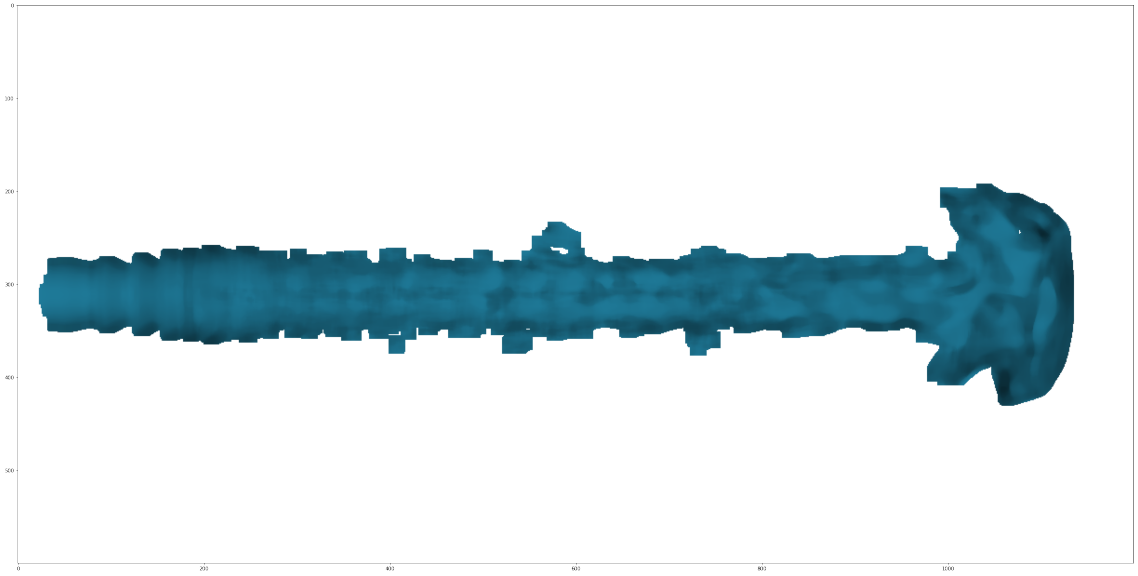

Figure 3.4: Synthetic main flow using median filter.

For the first problem, currently I can't come up with a complete solution other than specifying a pasting position based on the coordinates of the fluid flow. This trick can only be applied to a specific frame, but cannot be generalized because the position of the fluid flow changes over time. However, I still stick with it, because the final goal is to generate data so that deep learning can learn the context.

For the second problem - seamless pasting. I was introduced to an algorithm called Poisson blending. This is an algorithm based on gradient preservation to make it smoother to paste one source image onto a target image. However, my problem is much more complicated. If the target and source images have very different backgrounds, then poisson blending tends to give not very good results. And if the patches of the target image are very different on the 2 sides of the pasting position, then poisson blending tends to give poor results as well. Unfortunately, in our case, the attached ligaments are partly on the blue patched of the main flow, partly on white background, and they are very different. This cause the attached ligament either too faint or having seam.

I've been trying to eliminate those negative effects on Poisson blending, but the result is somehow not very satisfying. In the last attempt, the attached ligament must compromise between the seamless boundary and the preservation of color. The final synthetic image is as in Figure 3.5.

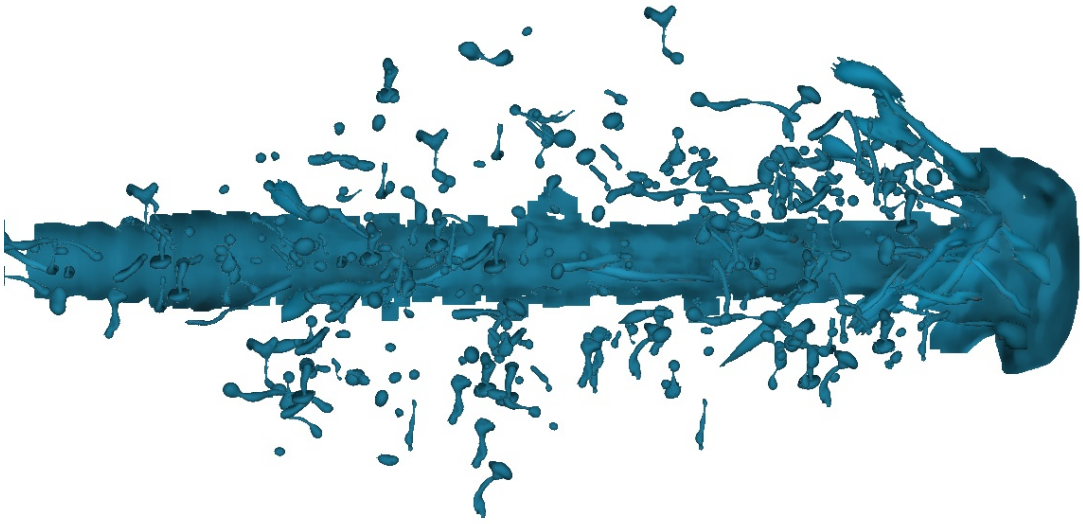

Figure 3.5: Final synthetic image for deep learning.

Chapter 4

Result and conclusion

I have been training a YOLOv5 model with 1500 training synthetic images and validating with 150 synthetic images. Each image consists of 150 drops and detached ligaments in totals, 15 attached ligaments. After 20 epochs, the mAP@0.5 scores reached 0.88. Figure 4.1 describes prediction of YOLOv5 on a synthetic image. However, the result on the final image is not very good (Figure 4.2). The prediction is quite noisy especially in dense places. The model produces many false positive ligaments. The reason could be that the synthetic data does not look like real image, especially ligaments. Actually, I have not augmented detached ligaments yet, and have not synthesized detached ligaments well. A neural network learnt from a set of data will not perform well on different set of data with a very different distribution. I believe the knot of the problem is that the distribution of droplets is so dense and the synthesized attached ligaments is clearly looking unreal. At the moment, actually my code is reasonably simple, and the techniques I have used so far is quite basic. In the future work, I will study more complex image processing techniques, more advance approach in deep learning, and make another attempt.

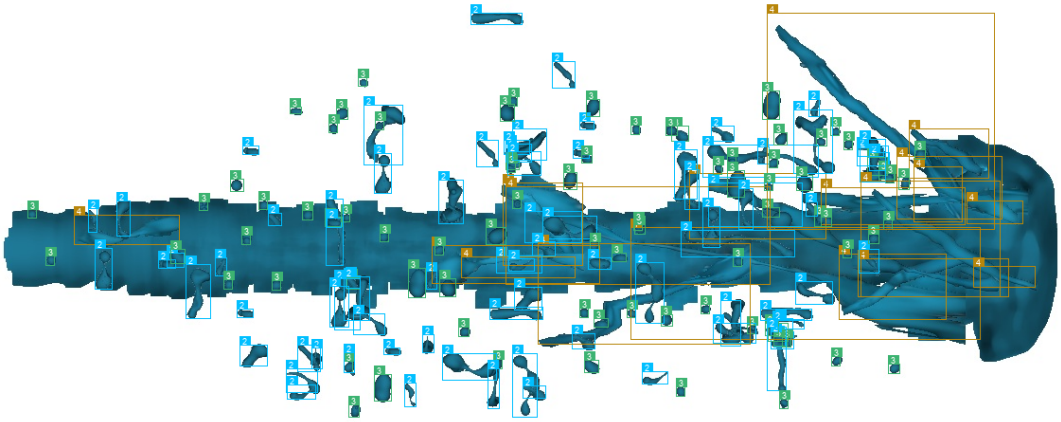

Figure 4.1: Prediction on synthetic validation image.
Green bounding boxes indicate drops, blue bounding boxes indicate detached ligaments, yellow bounding boxes indicate attached ligaments.

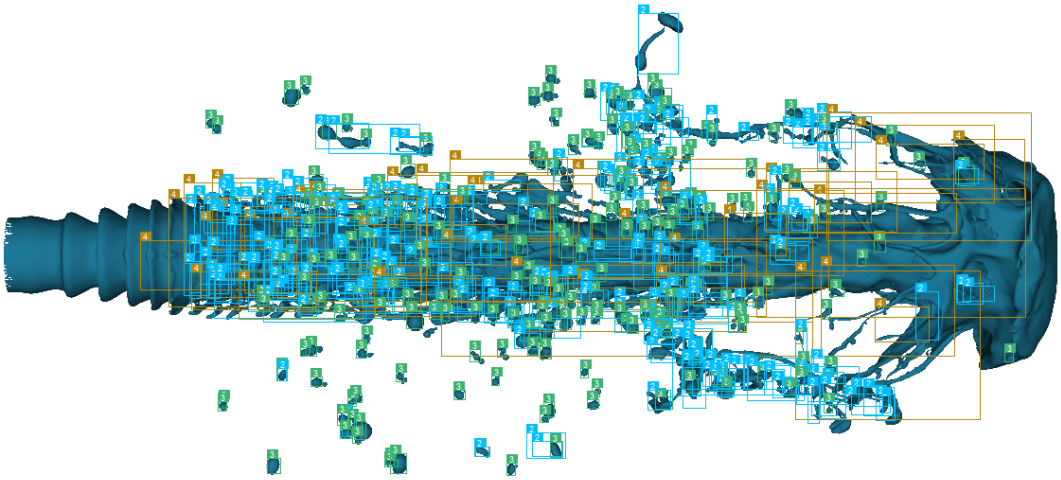

Figure 4.2: Prediction on synthetic real image.