Current Progress, Problems and future Ideas regarding Ecoli-Tracking

Elias Pinter

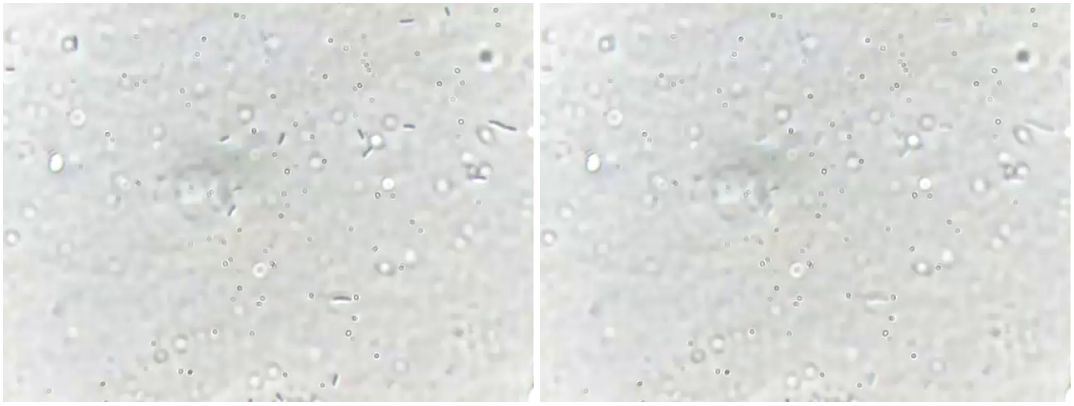
September 25, 2023

1 Detection

The approach that i have been using for detection actually does not use machine learning at all as it mainly exploits the fact that the background of a typical recording does not change. It is semi-automatic since it requires that the first frame of the video is labeled by hand. The Ecoli are then removed from the first frame via inpainting and some image processing to get a clean picture of the background. This image of the background can then be used to subtract from it from any frame in the video to obtain the Ecoli with some more image processing. The two code snippets **Alogrithm 1** and **Alogrithm 2** below describe the creation of the template and the detection process in an abstract way in pseudo code notation.

Algorithm 1 Creates a template by ipainting the user labeled Ecoli and blurring the neighbouring regions

for label **do**
    $x, y, w, h \leftarrow label$
    Store label for tracking
    $temp\_mask \leftarrow gray\_image[y : y + h, x : x + w] < threshhold$ ▷ Obtain Ecoli via Thresholding
    $mask[y : y + h, x : x + w] \leftarrow temp\_mask$
$template \leftarrow inpaint(first\_image, mask, radius)$
for label **do**
    $x, y, w, h \leftarrow label$
    $roi \leftarrow template[y : y + h + increase, x : x + w + increase]$
    ▷ Blurring is performed around the inpainted region to smoothen the image
    $GaussianBlur(roi)$
return template



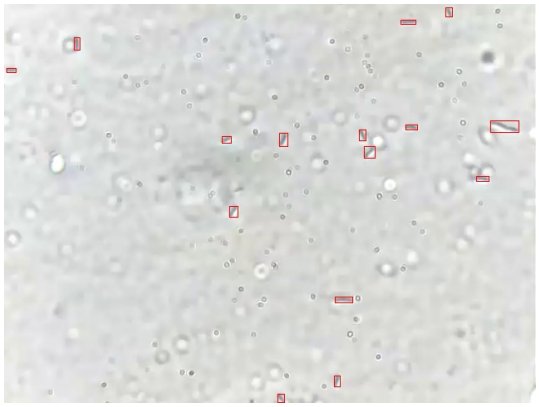
(a) Original Image (b) Template

Figure 1: Template

Algorithm 2 Detects Ecoli in a frame of the video using the template

$boxes \leftarrow [\ ]$
$subtraction \leftarrow subtract(template, image)$
$subtraction \leftarrow grayImage(subtraction)$
$ecoli\_noisy \leftarrow thresholding(subtraction, threshold)$
$ecoli \leftarrow medianBlur(ecoli\_noisy, kernel\_size)$
$contours \leftarrow findContours(ecoli)$
for contour **do**
    $x, y, w, h \leftarrow boundingRect(contour)$
    if $w \leq width\_cutoff \vee h \leq height\_cutoff$ **then** continue ▷ Skip boxes that are too small
    $x \leftarrow x - 1$ ▷ Make up for subtraction
    $y \leftarrow y - 1$
    $w \leftarrow w + 2$
    $h \leftarrow h + 2$
    $boxes.append((x, y, w, h))$
return boxes

(i) Subtraction (ii) Threshholded Image



(iii) Detections

Figure 2: Detection Process

2 Tracking

The current tracking approach is IoU based and makes use of domain knowledge about the ecperimental setup. The algorithm is based on the following four assumptions:

- Ecoli stay in there local environment since there are no cuts in the video
- Ecoli can only appear on the edge of the window
- Ecoli can only disappear on the edge of the window
- Ecoli can only seperate and never merge

Since it is not guranteed that our detection obeys these assumptions we need to know when certain events happen and handle them explicitely in our tracking algorithm.

2.1 Definitions

For describing the tracking algorithm we first need to through a couple of definitions. We will first start with bounding boxes. There will be two different ways of referring to bounding boxes dpepending on whether the Id of the bounding box is known. Id the id is not known yet we will refer to the boxes via indices.

$$b_t^i$$

1: Box with Id i at the frame t

$$b_{t,k}$$

2: Box with yet unknown Id with index k at the frame t

For the general tracking and to recognize if a certain event happened we will use two metrics, which we will also define, namely IoU(Intersection over Union) and IoA(Intersection over Area). Let $A(x)$ denote a function that returns the Area of a rectangle then we define for two boxes x and y

$$IoU(x,y) := \frac{A(x \cap y)}{A(x \cup y)}$$
$$IoA(x,y) := \frac{A(x \cap y)}{A(y)}$$

2.2 General Mapping and Important Events

First of all we will define the general mapping strategy to map boxes with Ids from a frame t to the boxes without Ids in frame $t+1$. We will map a box b_t^i to the box $b_{t+1,k}$ if $b_{t+1,k}$ maximizes the IoU score for b_t^i and is above a certain threshold. Let the general mapping be called f then we can define it in the following way:

$$f\left(b_t^i\right) = b_{t+1,k'} \text{ where } k' := \arg\max_k \ IoU\left(b_t^i, b_{t+1,k}\right) \text{ if } IoU\left(b_t^i, b_{t+1,k'}\right) > threshold \tag{1}$$

When tracking there are a couple of important events that we need to recognize. Those are

i. sudden appearance of cell

ii. sudden disappearance of cell

iii. merging of boxes

iv. cell duplication

the following statments describe how we will detect that one of those events happened. For i. we will say that a box at time $t+1$ appeared if there is no box at time t that maps onto it. So we say that box $b_{t+1,k}$ just appeared if fulfills the follwoing statement.

$$\nexists i \; f\left(b_t^i\right) = b_{t+1,k} \tag{2}$$

Similarly for ii. we say that a box b_t^i at frame t disappeared if there is no box in frame $t+1$ it maps onto. This means b_t^i disappeared if ot fulfills the statement

$$\nexists k \; f\left(b_t^i\right) = b_{t+1,k} \tag{3}$$

For event iii. we say that two boxes merged into a box at frame $t+1$ if they both map onto to the same box under f. This means that boxes b_t^i and b_t^j merged into a box $b_{t+1,k}$ if the follwoing statement holds

$$f\left(b_t^i\right) = f\left(b_t^j\right) = b_{t+1,k} \text{ with } j \neq k \tag{4}$$

Note that this statement can easily be expanded to more then two boxes.
For cell duplication we say that it took place if there are two boxes $b_{t+1,j}$ and $b_{t+1,k}$ whose IoA score relative to a box b_t^i is above a certain threshhold.

$$\exists j,k \; IoA(b_t^i, b_{t+1,j}) > threshold \wedge IoA(b_t^i, b_{t+1,k}) > threshold \text{ with } j \neq k \tag{5}$$

Now we have the fundamental building blocks for the tracking algorithm that is decribed below in **Algorithm 3**. For finding boxes that disappeared or were part of a merge we will perform a sliding window search with it's old image in the neighbourhood of its old box in the new image. It is also imortant to mention that we will only search for disappeared boxes if they were not close to the border and that we will ignore boxes that were not close to the border when they appeared. For the first frame of the video every box from the user labeling gets assigned an Id.

For the following Algorithm let $\max(I)$ denote denote a function that returns the highest previously assigned Id. B_t will describe the set of boxes at time t.

Algorithm 3 Tracks Ecoli between frames t and $t+1$

$detections \leftarrow detect(image_{t+1})$
$mapping \leftarrow \{\ \}$
$duplicated \leftarrow \{\ \}$
$merged \leftarrow \{\ \}$
for $b_t^i \in B_t$ **do**
    for $b_{t+1,k} \in detections$ **do** ▷ Calculate all pairwise metrics
        $IoU(b_t^i, b_{t+1,k})$
        $IoA(b_t^i, b_{t+1,k})$
    ▷ Check if box duplicated
    if (5) for b_t^i **then**
        $new\_id \leftarrow \max(I_t) + 1$
        $B_{t+1} \cup b_{t+1}^{new\_id}$
        $new\_id \leftarrow \max(I_t) + 1$
        $B_{t+1} \cup b_{t+1}^{new\_id}$
        $duplicated \leftarrow duplicated \cup i$
if $\max_k IoU\left(b_t^i, b_{t+1,k}\right) < threshold$ **then** continue
for b_t^i that statisfy (4) **do** ▷ Find merged boxes with sliding window search
    $new\_box \leftarrow sliding\_window\_search\left(b_t^i\right)$
    $B_{t+1} \leftarrow B_{t+1} \cup new\_box$
    $merged \leftarrow merged \cup i$
for $b_{t+1,k}$ that statisfy (2) **do** ▷ Check if boxes appeared close to the border
    if $distance\_to\_border(b_{t+1,k}) < threshold$ **then**
        $new\_id \leftarrow \max(I) + 1$
        $b_{t+1}^{new\_id} \leftarrow b_{t+1,k}$
        $B_{t+1} \leftarrow B_{t+1} \cup b_{t+1}^{new\_id}$
for b_t^i that statisfy (3) **do** ▷ Check if boxes disappeared far from the border
    if $distance\_to\_border(b_t^i) \geq threshold$ **then**
        $new\_box \leftarrow sliding\_window\_search\left(b_t^i\right)$
        $B_{t+1} \leftarrow B_{t+1} \cup new\_box$
▷ All old boxes that were not merged or duplicated get mapped according to (1)
for b_t^i where $i \notin (duplicated \cup merged)$ **do**
    $B_{t+1} \leftarrow B_{t+1} \cup f(b_t^i)$

This algorithm can be performed for successive frames until the last frame of the vidoe has its mapping. In the source code which can be found here. There is also a modified version of the BioPython library being used to create a population forest of the tracked Ecoli. If you have any questions regarding the source code just contact me on Discord: elson1608.

3 Experimental Setup

3.1 Problems

The problems we faced regarding the experimental setup are the follwoing

- Channel dries out
- Focus Drift

For the first one we first tried to stick two cones filled with LB in the two holes of the microfluidic channel. This did not really work since LB seems to be too thick. However when we tried the same with water it worked quite well. The only interesting thing about the water solution was that many Ecoli got ”gravitated” towards the two holes.

For the Focus drift we haven't really found a solution yet there are two causes that we suspect to be responsible for the loss of focus

- Temperature Fluctuations
- Stage Drift

Regarding the temperature fluctuations we measured the room temperature over the weekend and it seems like there is a difference of one degree Celsius between day and night as can be seen in **Figure 3**.

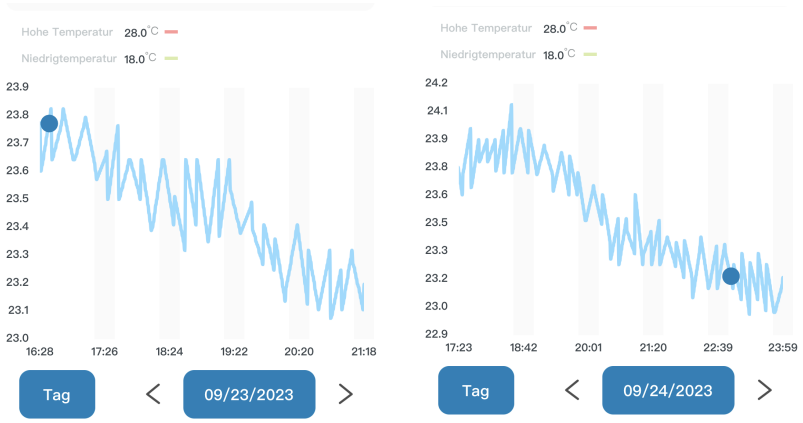


Figure 3: Temperature Fluctuations

Something else that we are trying to find out is during which stage in their lifecycle the Ecoli have the darkest color, since a high contrast is very beneficial for detection.

4 Possible Ideas for future Work

Regarding future work the most important thing will probably be to increase the quality of the videos and tackle the problems mentioned in **3**. Also one could try to find an alternative to the sliding window search that works better. Using the Watershed Alogrithm in very large boundg boxes during the detection stage could help at preventing merges and spotting duplication earlier. Also it could be interesting to look at mixed scenarios with Algae and Ecoli together.