Automated Analysis of Videos Depicting the Interaction Between Tuberculosis Bacteria and Immune Cells

Ida Bjerwe, Johan Edstedt, Oskar Löwek, Robert Cranston, and Therese Luong
Linköping University, December 2019

Abstract—We present several novel methods for analysis of microscopy videos of cell interactions. Specifically our main focus has been in the following domains,
(i): Tracking algorithms for collecting trajectory data as well as infection and cell death detection in macrophages. This is done using a combination of well known concepts in computer vision such as the optical flow algorithm by Lucas–Kanade and a state of the art convolutional neural network used for biomedical image segmentation, U-Net.
(ii): Image processing and clustering algorithms for detection and analysis of macrophage aggregation. Several properties are calculated, such as position, size and shape metrics.
We quantitatively and qualitatively show that our algorithms perform well on real data using multiple evaluation metrics. The average object tracking error for the Lucas-Kanade tracker was 4 pixels and it successfully tracked 74% while the assignment method successfully tracked 60% of all automatically cells throughout 100 frames.
Aggregation is detected correctly in 96% of 851 tested images.

1 Introduction

Recent advancements in technology have made it possible to generate a large amount of data from microscopy images of cells. This requires automated analysis of the data that is generated since it is very time consuming for scientists to go through this amount of data image by image. Another advantage of automated analysis is that cell behaviours can be visually enhanced in various ways and different statistics can be calculated. Maria Lerm together with her project group at the University hospital in Linköping are working on discovering opportunities for prevention of tuberculosis (University, 2019). They produce a large amount of microscopy image data and to be able to analyze the data in a more efficient way M. Lerm reached out to Linköping University to get help from students with image processing knowledge. This created a cooperation with the aim to find automated ways of analysing their specific data. An example on how their images look like is shown in Figure 1.

In this project we decided to focus on two domains where the first domain was tracking algorithms for collecting trajectory data as well as infection and cell death detection. Tracking of cells can be used for example to visually enhance how the cells are moving over time. The challenge with tracking cells instead of tracking e.g. cars is that cells change shape over time, a much lower framerate, and the fact that all cells look very similar. All this entails that a tracking algorithm might have difficulties to recognize the same cell in one frame to another.

The second domain was to use image processing and clustering algorithms for detection and analysis of cell aggregation. Cell aggregation is when cells move close to each other and form 'cell clumps' and identification of this cell behaviour in an automated way saves significant efforts for biologists when they want to analyze that specific behaviour.

This report describes the different methods used to make an automated analysis of these two domains as well as the result of the implementation.

2 Problem Formulation

The client wanted an image processing based tool for automatic analysis of microscopy image sequences of macrophages in contact with tuberculosis infection. What areas to specifically focus on

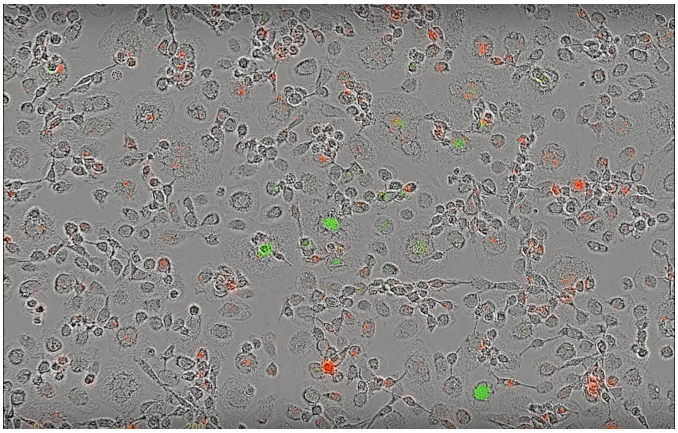


Figure 1. A microscopy image from M. Lerm and her project group including all three colour channels. The green in the image is bacteria, the red is cell death and the gray spots are cells.

was discussed in upcoming meetings, and was determined based on their priorities and our knowledge of limits in image processing. We narrowed it down to two main tasks.

Cell-to-cell spread of infection, which is the ability of bacteria that reside in the cytosol of the cell to spread to the cytosol of another cell without traveling through the extracellular medium (Kuehl et al., 2015). This was the end goal for this task, and a major step to solve this was to implement a tracker that can follow and store data of wanted cells, so that was decided as the main goal. The tracker should also create additional statistics about the cell that in future work can be of help for solving the original problem to detect cell-to-cell spread.

The other task was to detect cell aggregation in images, which is a recently observed phenomenon where cells aggregate, which is unusual for macrophages (Otaka et al., 2014). The goal was to be able to detect in which images this occurs and give statistics for further analysis.

3 Background

In recent years the biomedical field has undergone a transformation, with an explosion in the usage of data analysis methods like computer vision and machine learning techniques to improve results and be able to process the large amounts of data being generated by the industry (Dinov, 2016). In this section some previous work within automated analysis in the biomedical field will be presented. We first give a short introduction to microscopy imaging, cell staining, and *Mycobacterium tuberculosis* (Mtb) interaction with macrophages.

3.1 Microscopy imaging, staining methods and tuberculosis

Microscopy imaging, and specifically live cell imaging developed in the 20th-century. One of the major issues with live cell imaging is that cells are generally translucent, and hence difficult to observe with standard bright-field microscopes. Phase-contrast microscopy (Zernike, 1942) was invented in the 1940s specifically to deal with these issues. Instead of observing the light intensity, the phase shift of the light passing through the cells is observed and then converted into pseudo-intensities in the resulting image, which will henceforth be referred to as *phase image*.

There are several ways to make cells fluoresce. In this study, a genetically modified version of the mtb bacteria which fluoresces in a green hue was used. This is a preferred method over other invasive staining methods which damage and can even kill the cells (Ge et al., 2013). Another dye that has been used is a red dye which permeates the macrophage cell walls only in connection with cell degradation and death. A third dye, which makes the cell nuclei fluoresce has also been used, mainly in connection with the training of a cell segmentation network. As mentioned previously this kind of dye is toxic and should be avoided.

Many interesting dynamics have been observed between human macrophages and mtb. Macrophages, whose regular role in the human immune system is to digest and remove any unwanted foreign objects in the body, e.g. bacteria, are unable to properly digest mtb and are instead infected and serves as a 'breeding ground' for the bacteria which then can spread further in the body Ernst, 1998. Some of the questions which Maria Lerm's group at LiU are investigating are different phenomena which have been observed between macrophages and mtb. One such phenomenon is aggregation of macrophages, which usually are solitary cells, in connection with infection. Another is changes in cell behaviour after infection, e.g. the speed of the cell. Both of which were also discussed in the problem formulation section. It is with this in mind we developed our methods.

3.2 Previous work

Some examples of recent areas of success is medical image segmentation with U-Net (Ronneberger et al., 2015) outperforming all previous methods within lung cancer detection (Ardila et al., 2019),

and smart-watch detection of heart disease (*Heart rate notifications on your Apple Watch* 2019).

Automated analysis has also become more prevalent in tuberculosis research, Mahamed et al. (2017) used tracking in conjunction with infection and death data on cells to quantify several important statistics, such as time of infection, size of infection, and time of death. However, very few cells were tracked (around 40) in total, and to the best of our knowledge their algorithm are limited to single cell tracking.

Our research has focused on two main areas, *cell tracking* and *cell aggregation*.

There are several paradigms for tracking. To mention a few: Discriminative based approaches, Background Subtraction approaches, Point Tracking approaches, Generative approaches, and Dense/-Local Optical flow approaches (Satrughan Kumar, 2016; Dorra Riahi, 2016).

In microscopy images of macrophages there are several constraints that limit our choice of method. One such constraint is that all the cells are fundamentally of the same type. This means that Dense Optical flow and Discriminative methods have a high failure rate since it is often difficult to discriminate between objects in the image. Since the image is often filled with a large amount of cells this can be fatal. Another issue is that there is often a very small amount of static background, this means that background subtraction often produces incorrect backgrounds and may therefore give poor results.

For cell aggregation there exist several different strategies to access the aggregation feature. In Mahamed et al. (2017) the problem was approached by first determining the cell borders and then using the watershed algorithm to discriminate adjacent cells. The distance between the segmented cells is then calculated. Due to different setup, cameras that are used and different cell culture plates the images vary widely and make it challenging to find a parameterization that fits the whole sequence. Other approaches uses machine learning and deep learning algorithms to train a classifier and classify the detected objects (Sommer and Gerlich, 2013; Kraus et al., 2017). The limitation of machine and deep learning approaches is that it requires large amount of data and the parameters needs to be tuned in each step (Kraus et al., 2017).

4 Data

This section describes the data collection of the used microscopy images from M. Lerm and also the processing pipeline employed in the project.

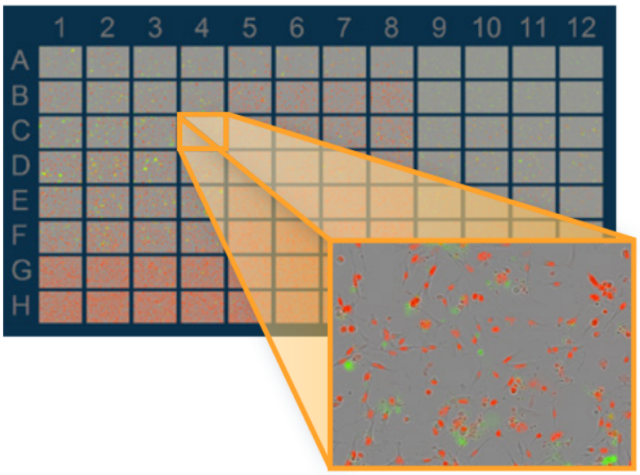


Figure 2. Example of visual interface of IncuCyte. Wells are ordered in a grid. A combined color image for a specific well at a specific time is magnified.

4.1 IncuCyte Live-Cell Analysis System

Cells and bacteria are studied in vitro, using small sample areas called *wells*, shown in Figure 2.

The data is generated through an automated microscopy image capture system called IncuCyte® S3 Live-Cell Analysis System (*IncuCyte® S3 Live-Cell Analysis System* 2019), henceforth called IncuCyte. This system captures hundreds of wells, where each well contains a sample, concurrently by iteratively shifting a microscope over the samples. A visualization of the physical IncuCyte system can be seen in Figure 3.

Wells are arranged in a grid indexed with a letter and a number. Experiment parameters are fixed within a single well and the response in cell behavior to varied parameters can be studied by comparing behavior in different wells in the grid. Each well can further be divided into several *regions*, indexed with a number, on which the camera places itself to capture images. These regions are in general not contiguous, so no stitching can be performed to e.g. track cells between regions.

IncuCyte comes with on-board image processing and storage in the form of proprietary server software which is accessible via remote login.

4.2 Data collection pipeline

The on-board software has the ability to produce images from different wavelength bands which gives added description capabilities e.g. that stained cells can be easily identified. This allows us to

Figure 3. Picture of the physical IncuCyte system.

gather all the data that are needed: a raw grayscale image of the cells, see Figure 5, an image where only the bacteria are seen, see Figure 4, and also an image where the cell death is shown, see Figure 6, but other types of staining and dyes are also possible.

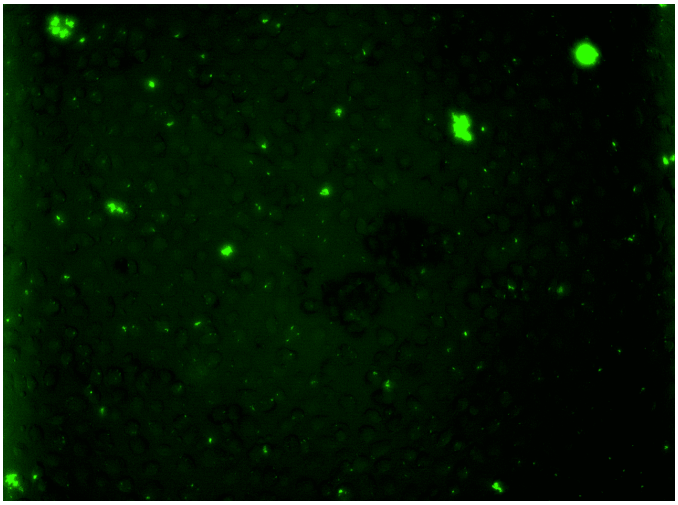


Figure 4. Green channel of microscopy image.

The number of cells, time resolution, and multiplicity of infection (MOI) (Ellis and Delbrück, 1939), i.e. number of bacteria per cell, are some of the sample parameters that need to be taken into account. In our case we typically deal with 100–500 cells per sample, a time resolution of between 30 minutes and 6 hours, and MOI of about 5–15.

In general, fewer cells and higher time resolution gives better tracking results. However this is not always possible, both for practical reasons and because setting these parameters too far in this direction has detrimental effects on the cells and their behavior. This has been one of the major

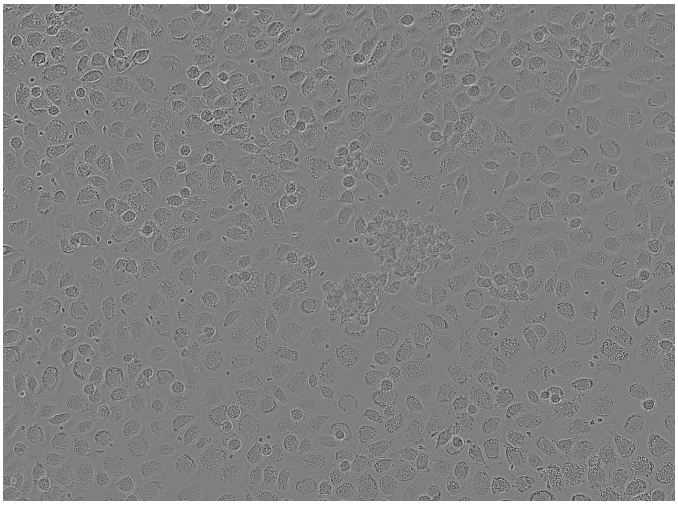


Figure 5. Phase channel of microscopy image.

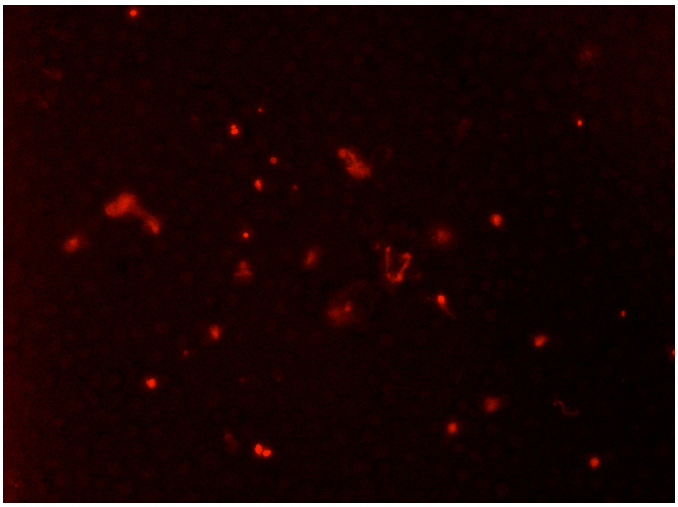


Figure 6. Red channel of microscopy image.

challenges in this project.

4.3 Storing/Loading

As discussed above, the data can be accessed with remote login to the IncuCyte server. However, this can only be done manually through a graphical interface.

To allow our software to read (and write if needed) the data the built in export functionality was used to download it locally. It was then uploaded to a Raspberry Pi (*Raspberry Pi* 2019) equipped with extra storage owned by the group from where it was served over the SSH File Transfer Protocol (SFTP) (*SSH File Transfer Protocol* 2019) by OpenSSH (*OpenSSH* 2019).

Public key cryptography was utilized to allow secure and automated access by the group members

only. SSHFS (*sshfs* 2019) was then used to mount the file store on the machines running the software, obviating the need to bake support for network access into the application while still allowing it to process data larger than what will fit on local storage. GNOME Virtual file system (GVfs) (*GNOME Virtual file system (GVfs)* 2019) was also evaluated for this purpose but was discovered to exhibit bugs such as inability to create symbolic links or delete files.

5 METHOD

In this chapter the project framework is described followed by the different methods used for each of the project tasks. Also the two different evaluation methods are described.

5.1 Project framework

This project used Scrum [1] as framework where a product backlog initially was formed followed by a number of sprints that were updated each or every second week. The framework focuses on smaller milestones instead of bigger milestones which makes it easier to know what needs to be done in a shorter amount of time. Each task in every sprint was marked with 'To do', 'Doing' or 'Done' and also who the different tasks were assigned to. That gave a good insight in the development of the project for all project members. Beyond this, we had a meeting with the customer M. Lerm and her project group once every other week on average.

5.2 Tracking

Tracking multiple objects with very similar features turned out to be a major challenge for traditional tracking approaches. It was quickly realised that discriminative approaches for trackers were unstable. Traditional dense optical flow approaches failed and more modern approaches such as SpyNet (Ranjan and Black, 2016) failed which we conjecture is caused by the domain shift from natural images to microscopy. Hence both these approaches were discarded. We ended up utilizing two approaches, which both used our cell detection network in varying ways.

The first approach uses optical flow to estimate the new position of the cell between frames, and also segments this next frame to get the cells and with some image processing also the positions where the cells are located. First we begin by a high level description of the tracking procedure utilized in both methods, which is presented in algorithm 1.

The block diagram in Figure 7 gives an overview of the different steps used in the tracking part of this project and will be described further in upcoming sections.

input : ϕ_i: a phase image
input : TB_i: an uncalibrated tuberculosis image
input : D_i: an uncalibrated cell death image
input : $M\Phi_{0:i-1}$: cell traces for tracked cells until previous frame

$TB_i \leftarrow$ `calibrateImage(`TB_i`)`
$D_i \leftarrow$ `calibrateImage(`D_i`)`
$\delta \leftarrow$ `detectPerturbation(`ϕ_i,ϕ_{i-1}`)`
ϕ_i,TB_i,$D_i \leftarrow$ `shiftImage(`ϕ_i,TB_i,D_i`)`
$M\Phi_{iU} \leftarrow$ `detectCells(`ϕ_i`)`
$M\Phi_{0:i} \leftarrow$ `track(`ϕ_i,ϕ_{i-1},$M\Phi_{iU}$,$M\Phi_{0:i-1}$`)`
$M\Phi_{0:i} \leftarrow TB_i$,$D_i$

Algorithm 1. General formulation of tracking.

5.2.1 Perturbation adjustment

A quirk of the data collection, described in more detail in Section 4, is that images in the sequences were often not aligned perfectly, and in fact often contained large spatial perturbations. This greatly hampers the performance of any tracker, and hence we created an algorithm to automatically perform an inverse perturbation to restore temporal coherence to the sequences.

This was re-framed as a simple maximum correlation problem

$$P_t^{-1} = \arg\max_{\delta} \sum_x I_{t-1}(x) I_t(x+\delta). \qquad (1)$$

where position is denoted by x and displacement by δ. I_t is the current frame and I_{t-1} is the previous frame.

The underlying assumption is that the global alignment that corresponds to the camera perturbation gives the highest correlation, i.e. we assume that the cells in general do not move 'too much'.

Now, to make this process more stable some standard image processing techniques were used where the images were thresholded and blurred. To speed up computation the operations were conducted in the fourier domain which can be shown to be

1. Lecture: Introduction to scrum in course TSBB11. Klaus Tindholm. Linköping University 2019.

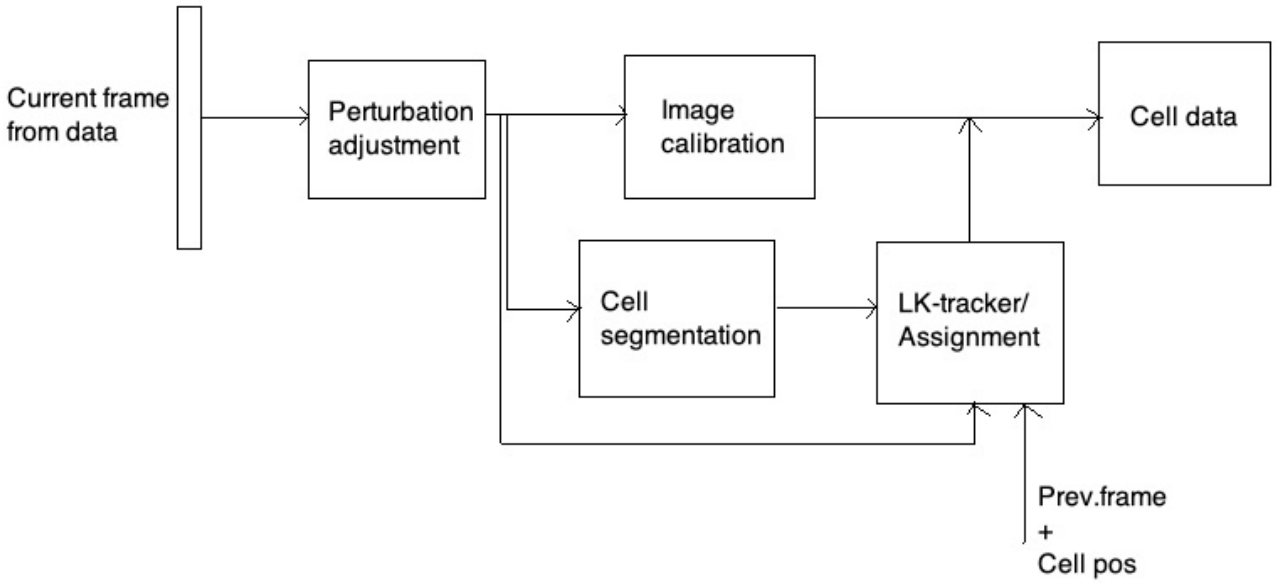
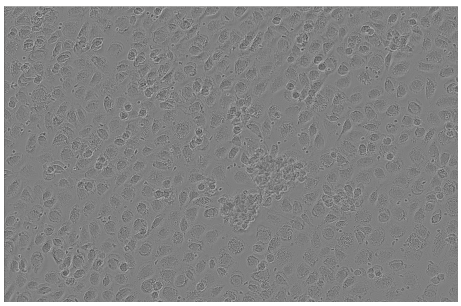
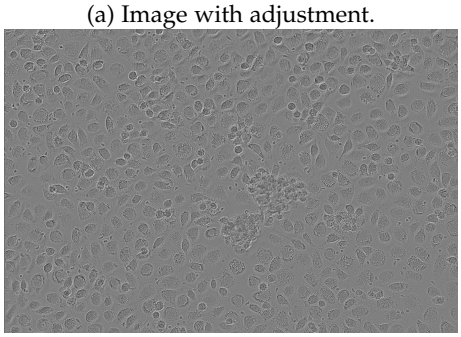


Figure 7. Block diagram for tracking method.

equivalent to the circular correlation. This technique turned out to work very well and can be seen in Figure 8.



(a) Image with adjustment.



(b) Image without adjustment.

Figure 8. Consecutive images overlaid.

5.2.2 *Automatic calibration of images*

The images produced by the IncuCyte system can either be exported as pre-calibrated or raw. For both our domains we chose to work with the raw images, to avoid potential issues resulting from changes in IncuCyte algorithms and also so that our method is not dependent on any extrinsic algorithms. While the phase images did not need much calibration, the red and green channels of the microscopy image are extremely noisy and need several calibration steps. The procedure is explained in Algorithm 2.

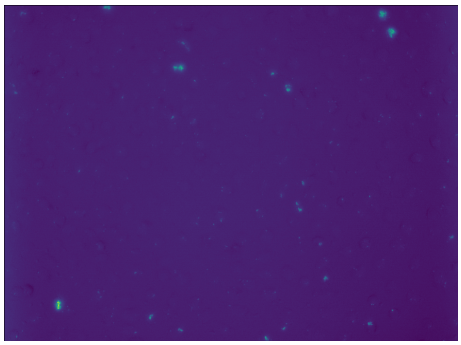
input : im, an uncalibrated image
output : im, a calibrated image
parameter: σ, standard deviation of gaussian blur
parameter: s, kernel size of maximum filter

$DC \leftarrow \texttt{smoothImage}(im, \sigma)$
$im \leftarrow \texttt{rectify}(im - DC)$
$\varepsilon \leftarrow \texttt{median}(\texttt{maxFilter}(im, s))$
$im \leftarrow \texttt{rectify}(im - \varepsilon)$

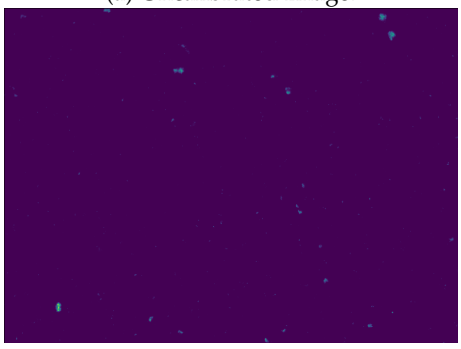
Algorithm 2. Image calibration.

Smoothing is done to remove local DC-level unrelated to signal. Here σ should be quite large, we used $\sigma = 50$. Then we rectify to remove the noise which we assume is negative, which means that we threshold at zero. We are still however left

with a certain amount of random noise in the image. We use the median local max-filters over the image to estimate an upper bound on noise amplitude. We then subtract that and rectify the image again. This produces our final result that can be seen in Figure 9. Observe that this procedure only works on images where we can assume that the signal plus noise has higher amplitude than noise. This is not necessarily the case, for example cell bodies can actually block certain amounts of background radiance and will therefore have a lower intensity in the image, however for our applications the signal can safely be assumed to be positive. Another assumption is that it is possible to estimate the noise with the median of a local maximum filter, this requires enough area of noise.



(a) Uncalibrated image.



(b) Calibrated image.

Figure 9. The obtained calibrated and uncalibrated images.

5.2.3 Cell segmentation

We implemented a cell segmentation network based on the U-Net architecture (Ronneberger et al., 2015). We trained the network using sequences where cell nuclei had been stained, and used this dye as output labels for the network. We trained on an Nvidia 1080 8GB GPU on 256x256 crops of images down-sampled to 260x352 in batches of 16 with an Adam optimizer with default settings and a exponentially decaying learning rate with a factor of 0.95.

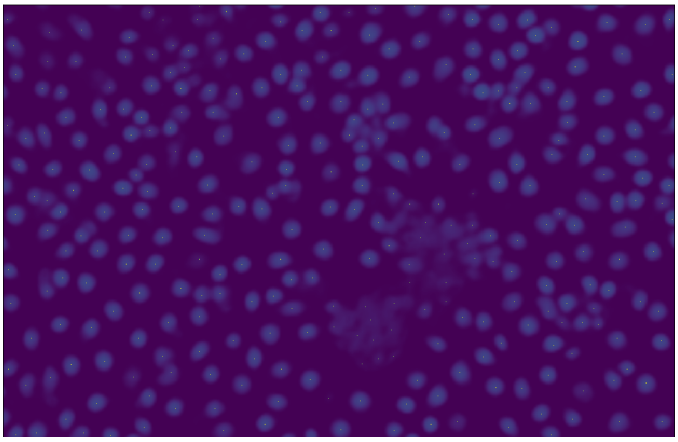


Figure 10. Sample outputs of the cell detection network.

Some label outputs for a select set of microscopy images of the network can be seen in Figure 10.

After detection, the detection image map was up-sampled linearly to the original image size.

It was observed that some data were unusable and damaging for the learning of the network. Since the labels were generated from the data itself issues such as late registration of cell dyes resulted in empty label images. Therefore sequences were manually selected based on visual inspection to ensure correctness of the learning.

One of the major advantages of this kind of segmentation compared to traditional thresholding or gradient filtering is that it is much easier to discriminate between nearby cells since only the nuclei were detected. This has proved very important for the stability of tracking.

5.2.4 Optical flow method

The method used to estimate the optical flow, which also can be seen as the motion, between frames is the one presented by Lucas-Kanade (Lucas and Kanade, 1981), and will be referred to as the LK-tracker. Optical flow assumes that the flow is essentially constant in a small local neighbourhood around the point under consideration, which is a correct assumption in this case since the point to be tracked should be chosen close to the center of the cell. However, this method is sensitive to large motion since it assumes that the motion of the point between two consecutive frames is small. To make this more robust to cases where larger motion occurs, a scale pyramid is used. The optical flow algorithm runs on all the scales of the image, where on the higher scales small motion is removed and large motion becomes small motion which the algorithm then can detect. The search window used

around the point of consideration is 15 times 15 pixels and the number of scales used is 4. Since this is an iterative method it also uses termination criteria in the form of maximum number of iterations and minimum search window displacement. Maximum number of iterations used is 10 and minimum search window displacement is 0.03 pixels. Consequently, the Lucas-Kanade tracker calculates the dissimilarity between two local regions in two images as

$$\epsilon = \iint_W [I_{t+1}(x+d) - I_t(x)]^2 w(x)\, dx \qquad (2)$$

where position is denoted by x and displacement by d. I_t is the current frame, I_{t+1} is the next frame, W is the integration region around a pixel and w represents the weighting function (Johansson, 2007).

Input : ϕ_i: Current phase image
Input : ϕ_{i-1}: Previous phase image
Input : ${M\Phi_i}^U$: All detected cells in current image
Input : $M\Phi_{0:i-1}$: Cell traces for all tracked cells up until last image
Output : $M\Phi_{0:i}$

${M\Phi_i}^{LK} \leftarrow$ `estimateFlow(`ϕ_i,ϕ_{i-1},$M\Phi_{i-1}$`)`
$M\Phi_i \leftarrow$ `fuseEstimates(`${M\Phi_i}^U$,${M\Phi_i}^{LK}$`)`
$M\Phi_{0:i} \leftarrow M\Phi_{0:i-1} + M\Phi_i$

Algorithm 3. Lucas-Kanade tracking method.

5.2.5 Assignment method

The second method is perhaps the most conceptually simple. If the segmentation in the cell detection output is reduced to points for two consecutive images in a sequence the tracking problem can be framed as an assignment problem where the global sum of travel-distances is to be minimized for all the cells, under the constraint that all cells in the first image must be assigned a cell in the new image.

From this we formulate a matrix containing all distances between all points in both images, on which we then employ the Hungarian algorithm (Kuhn, 1955), to assign each cell a new position. In a similar fashion as in the perturbation adjustment we have assumed that the cells do not move too much, i.e. that cells usually do not cross paths.

A weakness of this method is that it is completely point based, compared to the Lucas Kanade tracker which contains a priori information about the cell shape. A potential reframing of the distance matrix with consideration taken to appearance change could potentially be an interesting extension, however this might introduce issues since cells often look very similar. The procedure is presented in algorithm 4.

Input : ϕ_i: Current phase image
Input : ϕ_{i-1}: Previous phase image
Input : ${M\Phi_i}^U$: All detected cells in current image
Input : $M\Phi_{0:i-1}$: Cell traces for all tracked cells up until last image
Output : $M\Phi_{0:i}$

$C \leftarrow \sqrt{\sum(M\Phi_{i-1} - {M\Phi_i}^U)^2}$
$M\Phi_i \leftarrow$ `minimizeAssignmentCost(`C`)`
$M\Phi_{0:i} \leftarrow M\Phi_{0:i-1} + M\Phi_i$

Algorithm 4. Hungarian Assignment Method.

5.2.6 Cell data and visualization

Some additional data was obtained for the cells that were tracked. This data is shown and updated in the terminal as well as in an overlaid trace in an image (if wanted) as the program was running. The cell data consist of:

- Cell ID
- If the cell is dead, alive or infected
- Infection amplitude
- Distance the cell has traveled
- The time that a cell has been infected for
- Positions of the cell in every frame
- How many frames the cell was tracked in
- When the cell was infected
- Speed
- When the cell died

From the red and green color channels of the cell images from IncuCyte values on cell infection and cell death could be obtained. This was done by creating a box of a predefined size of 40 times 40 pixels which is the observed average size of a cell. The box was then placed around a chosen cell in the image and the amplitude was calculated as the sum of the values in the image divided by the area of the box. Two different thresholds, one for cell infection and one for cell death, were set to values that determine if the cell would be assigned as infected, dead, alive or not infected. This threshold can be changed by the user if desired. If a cell was assigned as infected the program starts to calculate for how

many frames the cell was infected and displays the statistics in the terminal. The visually showed trace also changes color from blue to green if a cell is assigned as infected. If a cell was assigned as dead the time of cell death was saved and displayed in the terminal and the current position of the cell was marked with a bigger red dot to show the user that the cell has died. Since the distance that a cell has traveled as well as the time resolution was known the speed of the cell could easily be calculated.

5.2.7 Program I/O

The finished developed program has some parameters that affect the execution, such as number of frames to run all the implemented algorithms on, thresholds for infection and cell death, if you want to output a movie of the sequence with an overlaid trace etc. These are examples of parameters that can be changed by the user and were implemented in a configuration file that the program reads from start. The configuration file is in JSON format.

The program outputs data from the run to a file. These data contain various data of the cells that were being tracked. These data can then be analyzed and printed in a user-friendly way by running a second script.

5.3 Aggregation

In this section an overview of the architecture and the methodology for detecting aggregation is presented. The user interface will also be described.

5.3.1 Overview

The system architecture for detecting aggregation is illustrated in Figure 11, where several modular building blocks are combined to form different methods. To find aggregations, two different approaches were used: a mask-based method using morphological operations and a point-based method with a clustering algorithm. Evaluation of the methods can be found in Section 7.2.1.

The function `image_to_mask` is the first step in both methods and converts the original image (phase image) to a binary image. The next step in the mask-based method is to label the connected components and return the found aggregations. This is handled by the `mask_to_aggregations` function. In the point based method, after the mask has been calculated, the `mask_to_positions_and_weights` function sets positions as the local maxima of distance-to-background and assigns weights to the

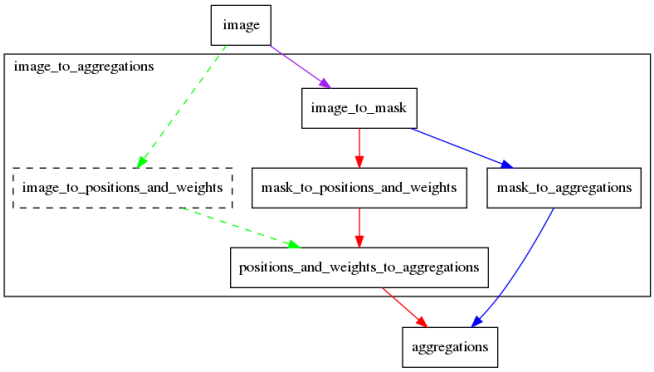


Figure 11. Overview of the architecture for aggregations. To find aggregations, two different approaches were used: a mask-based method using morphological operations (shown in blue) and a point-based method with a clustering algorithm (shown in red). A third point based method was planned, utilizing the trained U-Net model mentioned in the Section 5.2.3 (shown in dashed green). However, it is not yet implemented and is left for future work (see Section 9).

positions based on the distance. Then the function `positions_and_weights_to_aggregations` performs clustering and returns the found aggregations. These are described in more detailed in the coming sections.

5.3.2 Create a mask

The naïve way to generate the mask would be to calculate the value of the background, e.g. by assuming it to be the median, and setting values sufficiently far from it to true and the rest to false. As can be seen in figure 12 however, this produces less than satisfactory results.

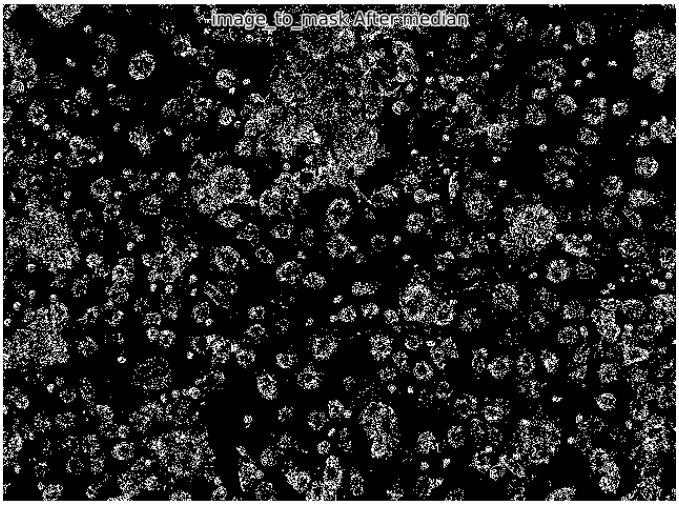


Figure 12. Image after thresholding on a band around the median.

To improve this, we note that pixel regions containing cells are characterized by varying more than the background. Therefore the image is differentiated by calculating the norm of the gradient using Sobel operators

$$|\nabla I| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}. \quad (3)$$

The result is shown in Figure 13. This is then thresholded with a threshold found by inspecting the histogram in Figure 14.

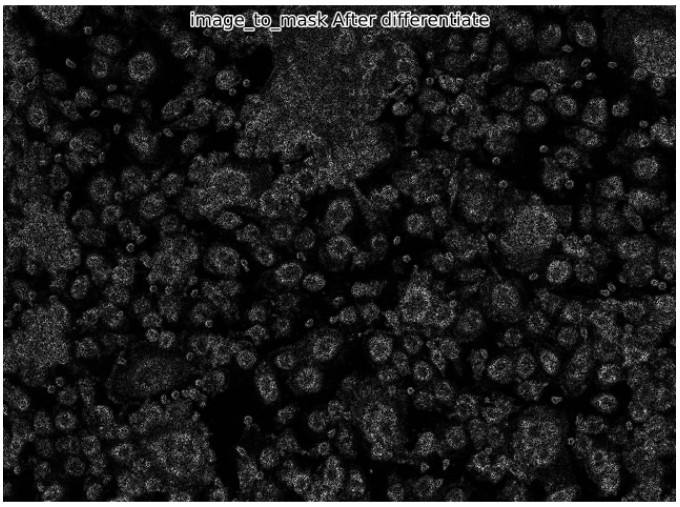


Figure 13. Image after differentiation.

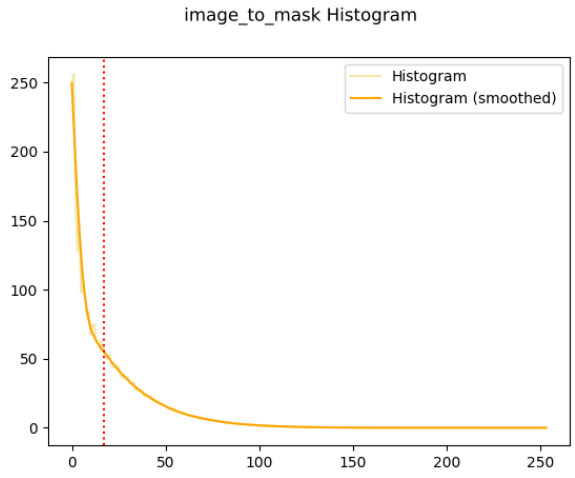


Figure 14. The histogram of the differentiated image with the calculated threshold highlighted.

The differentiated image is dominated by pixels with low values, corresponding to the background. We wish to place the threshold just to the right of this peak. To do this we calculate the derivative and set the threshold to the left-most point with a slope ≥ -2 (this value is the default and can be modified). However since the histogram and the derivative is not smooth this comes with problems. Therefore, a low pass filter is applied and the condition that the second derivative must be positive at the threshold point is added. This is illustrated in Figure 15.

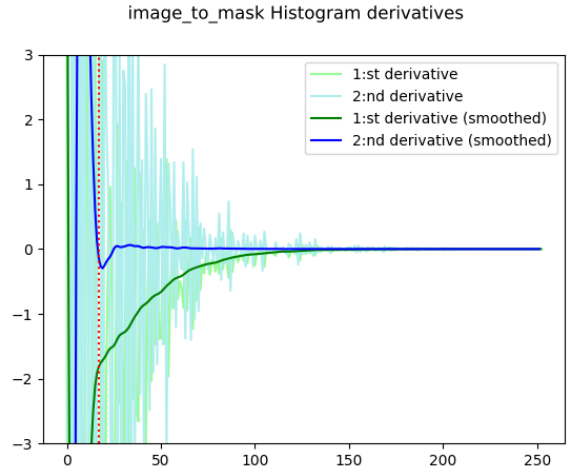


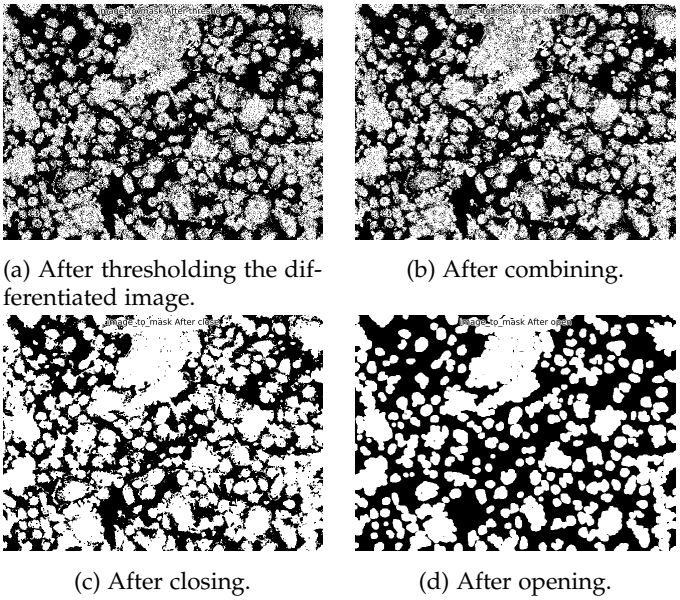
Figure 15. The first and second derivatives of the histogram of the differentiated image with the calculated threshold highlighted in red. Note the profound difference made by the low pass filtering.

The result, shown in Figure 16a, is combined with the naïve approach, Figure 16b and finalized with morphological opening and closing to remove noise and fill holes, Figures 16c and 16d, respectively.

5.3.3 *Find aggregations from mask*

In `mask_to_aggregations` the connected components of the mask are found and a number of statistics calculated:

- All pixel locations of the aggregation are averaged to the *position* of the entire aggregate.
- The number of pixels of the aggregate are summed to give the *size*.
- Principal component analysis (PCA) is used to find the lengths of the principal axes. The larger is divided by the smaller to yield a *shape ratio*.
- Even if the shape ratio is close to 1, there may be many holes in the aggregate. To measure this a circle of the same total size as the aggregate is created, the intersection with the mask is performed and the result is summed. This number is then divided by the size to give a *compactness* score.



(a) After thresholding the differentiated image.

(b) After combining.

(c) After closing.

(d) After opening.

Figure 16. Intermediate results during aggregation detection using the mask method.

All of these statistics can be used to discard aggregates that e.g. are too close to the image border, too small, or not 'clumpy' enough to be of interest.

5.3.4 Create positions and weights from mask

The `mask_to_position_and_weights` function tries to estimate cell positions from a mask. An illustration is shown in Figure 17. It does this by assigning as position the innermost point of contiguous blobs.

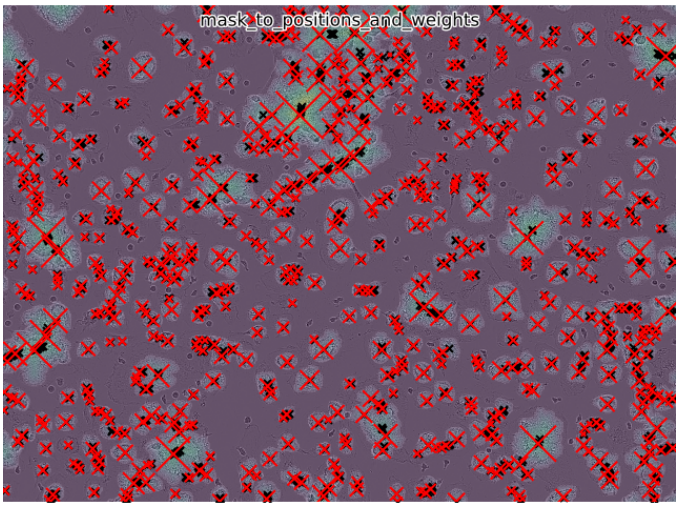


Figure 17. Intermediate result after `mask_to_position_and_weights`. Black markings are local maxima, red markings are local maxima after pruning and assigning weights.

To this end, the minimum Euclidean distance to the background is calculated for every pixel (bluish hue in Figure 17) and the local maxima found (black marks in Figure 17). Every maximum is then assigned a weight which is equal to the distance to the background. The uneven shape of aggregates and cells generates too many local maxima, therefore the maxima are pruned so that no two lie within a distance corresponding to each other's weight.

The surviving positions and the corresponding weights (red marks in Figure 17) are then returned.

5.3.5 Create aggregations from positions and weights

The `positions_and_weights_to_aggregations` function performs Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (**dbscan**), which is a clustering algorithm that clusters data and discards outliers. The algorithm takes in two parameters: `eps` and `minWeight`. The `eps` parameter is the maximum distance between two points for them to be considered neighbors and `minWeight` is the minimum sum of weights of points in the neighborhood for a point to be considered a core point. Core points then bind other points within the distance `eps` to the cluster as non-core points. Any remaining points are considered noise. The resulting clusters are considered aggregates.

The *positions* of the aggregates are set as the weighted average of the included point positions, and the *size* is set as the sum of the weights.

5.3.6 User interface

To make the system user-friendly a Command Line Interface (CLI) was used with the Python `argparse` module (*argparse — Parser for command-line options, arguments and sub-commands — Python documentation* 2019). It takes in a set of positional and optional argument and generates help messages. More about the CLI is described in the user manual.

6 Validation Metrics

In this section we present the quantitative and qualitative test metrics used for tracking and the validation metrics for aggregation.

6.1 Tracking

Four different evaluation methods were used to test the implemented tracker. Three of these test the quantitative robustness of the tracker and one tests the qualitative robustness. These four methods are described in detail below.

6.1.1 Quantitative evaluation

Since there was no ground truth available, an approximate ground truth was visually created to be able to test the implemented algorithms. For the first test that implies that one of us visually followed the cells that were evaluated and clicked in the images to save the coordinates for the new position of those cells in every frame. This means that the created approximate ground truth contains some errors that arise from not clicking on the exact same position in the cells through all frames. The visually detected coordinates were saved and compared with the coordinates that the implemented tracker detected for every cell that was evaluated. Both the ground truth and the predicted trace was drawn linearly between tracked points and might therefore not be the correct path between the points. To ensure that the program did not stop tracking a cell that was detected as dead during the tests the death threshold was set to a very high value during this process. The average Euclidean distance difference between the visually created coordinates and the coordinates from the tracker was calculated and analyzed as 4.

$$\frac{1}{N_g} * \sum_{i \in g(t_i) \wedge r(t_i)} \sqrt{(x_i^g - x_i^r)^2 + (y_i^g - y_i^r)^2} \qquad (4)$$

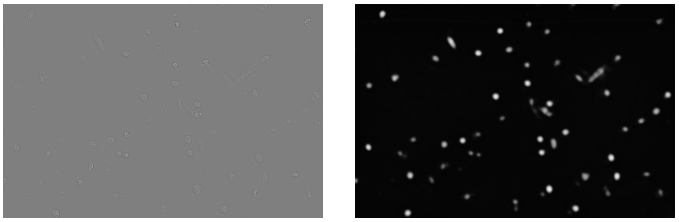
where N_g represents the number of frames that the cell was tracked, x_i^g represents the x-coordinate of the ground truth in the i^{th} frame, x_i^r represents the x-coordinate of the tracker in the i^{th} frame.

This is a good enough approach for quick and easy evaluation of the tracking algorithm and from now this distance will be called Object Tracking Error (OTE) (Bashir and Porikli, 2006).

The second quantitative test was to evaluate the number of cells that the tracker was able to track throughout all frames in a sequence. This test was needed because even if the earlier described test shows how accurate the tracker is it does not say anything about how many cells that the tracker lost throughout a sequence.

This was done by randomly choosing a number of cells to track in a sequence and visually inspecting the number of cells that were lost and determining if they were lost because they died or because the tracker did not recognize them anymore. If the cause of the lost track was cell death it was detected as a correct tracking but if the cause was that the tracker did not recognize the cell anymore it was detected as a failed. It was hard to determine a test limit for this test since there are no references on how many cells that the tracker can lose and still pass the test. Therefore this test only shows how many cells that the tracker lost.

A third test was also conducted on the cell detection algorithm which is a crucial part in the tracker as it removes the tracked cell if it can not be found in the detection image output. This test was done by manually determining where cell nucleus, which the detection algorithm is looking for, are located in the phase image (Figure 18a), for all cells visible. The same evaluation metric was applied on the cell detection image (Figure 18b). The number of cells in both images was counted and how many of them that matched was noted, with a match corresponding to points that are less than 20 pixels apart in both images. As the test above this was also done by manually labeling and setting the 'ground truth' in the phase image, which likely results in some error in the ground truth as well as in the labeling of the detection image.



(a) Cells in phase image. (b) Cells in detection image.

Figure 18. Difference in phase image and detection image

6.1.2 Qualitative evaluation

While a quantitative analysis is the preferred scientific method for assessing the quality of the results produced by an algorithm, there is in some cases also merit in a qualitative analysis. One such reason is that in many cases it is very difficult to generate ground truth traces to compare objectively. For example densely populated sequences may be extremely difficult even for a human to be able to properly label due to the high rate of misidentification between nearby cells. The distance between two traces can also sometimes be misleading, e.g. mistaken identity between two cells that move close together will give a low Euclidean distance, but obviously such a trace is not the desired outcome. To get a better intuitive understanding of the implemented algorithm, a selected number of cells were tracked in sequences that were sparsely populated, medium populated and densely populated and those were visually reviewed regarding misidentification.

6.2 Aggregation

6.2.1 Quantitative evaluation

In the same way as for the tracking data there was no ground truth given in our data for the aggregation problem. An approximate ground truth data was created by manually labeling data as *yes*, *no* or *do not know*. Ground truth was created for the following experiments: 'w49_2018' (121 frames), 'w8_2019' (219 frames), 'w6' (15 frames) and 'w4' (496 frames). To be considered as aggregation the aggregation should be clearly visible and have a total size larger than 12 average-sized cells in the image. In the evaluation our system is compared with ground truth and a ratio of correctly assigned labels is calculated. Images that is labelled in 'do not know' are not included in the evaluation.

7 Results

In this section we present quantitative and qualitative test results for the implemented tracking algorithm and the validation test results for aggregation.

7.1 Tracking

The quantitative and qualitative test results of the tracker are shown below.

7.1.1 Quantitative results

All results presented here are using the Lucas-Kanade tracking method described in Section 5.2.4. As discussed in Section 6.1.1 the object tracking error (OTE) between ground truth cell traces and predicted traces from the tracking algorithm was used as an evaluation metric. As this distance is dependent on the pixel resolution, it is of note that the average cell radius in the images was observed as about 30 pixels. Results are shown in Table 1 where three test sequences were used and ten different cells in every sequence were tracked. The test sequences that were used came from an experiment performed on 15th of November 2019 named *Clara few cells*.

The test results from Table 1 were also visualized by plotting the ground truth traces in a green color and the predicted traces in a red color. This made it easier to see how well the predicted traces agreed with the ground truth traces. Figure 19 shows the visualized result for the first tracked cell in sequence C4_2 from Table 1.

Another test on the quantitative result of the implemented tracker was, as mentioned earlier, the

Table 1
Test accuracy of tracking algorithm for three different sequences and thirty different cells. The letter and number that defines the test sequence stand for which well in that experiment that the images were collected from. Each well contained around five thousand cells from the start of the experiment.

Test sequence (50 frames/sequence)	OTE [Pixels]
C4_2	**4.43**
C4_2	2.63
C4_2	2.51
C4_2	4.14
C4_2	2.48
C4_2	6.08
C4_2	7.72
C4_2	4.60
C4_2	3.05
C4_2	3.67
B5_4	**4.07**
B5_4	2.62
B5_4	4.10
B5_4	3.31
B5_4	5.35
B5_4	4.63
B5_4	3.88
B5_4	5.48
B5_4	4.92
B5_4	4.66
B6_4	**4.07**
B6_4	2.62
B6_4	3.25
B6_4	3.82
B6_4	3.02
B6_4	2.86
B6_4	4.84
B6_4	3.46
B6_4	3.52
B6_4	3.92

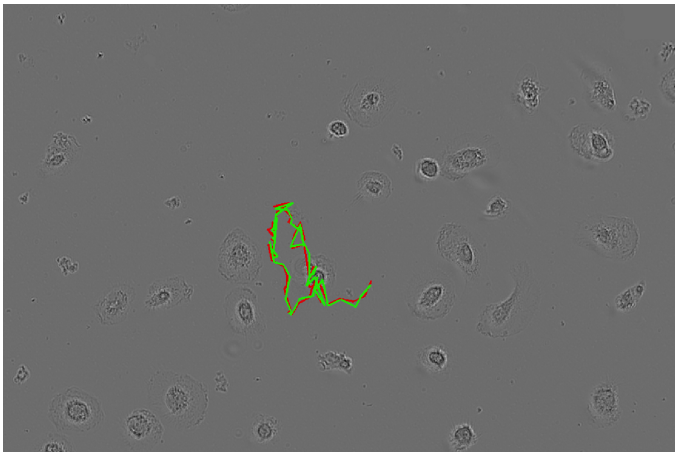


Figure 19. Example trace where ground truth trace is marked as green and predicted trace is marked as red (zoomed).

number of cells that the tracker was able to track throughout all frames in a sequence. This test was performed on the four different test sequences that were used in Table 1 and tracked in total 57 cells throughout the three sequences. The result can be seen in Table 2.

Table 2
Number of successful tracking instances compared to initial amount, C4_2, B5_4, B6_4, B2_1. Observe that the result for sequence B2_1 is in parenthesis. This sequence is more densely populated and difficult to track and evaluate.

Test sequence (100 frames)	LK	Assignment
C4_2	13/19	10/19
B5_4	13/17	11/17
B6_4	16/21	13/21
B2_1	(11/49)	(26/49)
Total test result	42/57(11/49)	34/57(26/49)

The third test mentioned in Section 6.1.1 was done on 11 randomly picked images from three different sequences where 5 of them had a bit fewer cells than the remaining 6 with the results shown in Table 3.

Table 3
Phase and detection image cell count comparisons. It shows that the detection algorithm finds more detections than was manually labeled from the phase image. Although the most important result that can be seen is that it on average finds 96.9% (770/795) of all cells labeled as ground truth and most likely to be chosen to be tracked.

Test sequence	# cells (phase, detection)	# cells (match)
C4_2	56, 61	54
C4_2	56, 66	54
C4_2	53, 60	51
C4_2	44, 51	42
C4_2	43, 54	42
B5_4	105, 118	103
B5_4	95, 104	92
B5_4	79, 94	78
B6_4	88, 101	85
B6_4	87, 101	82
B6_4	89, 108	87

7.1.2 Qualitative results

Here, three image sequences with different amount of cell density are presented. For the low density sequence, Lucas-Kanade tracking was used, while the assignment method was used for higher densities.

In Figure 20 an almost perfect tracking is observed of the cells with high precision. Some cells are immediately lost at the start of the sequence and their traces were discarded, which may indicate a slight selection bias for track able cells.

In Figure 21 a similar result can be seen, with most cells being traced correctly. In Figure 22 a slight degradation of the tracking can be seen, when the cells get very close to each other the tracker occasionally switches identities which indicates that results on such sequences are not entirely trustworthy.

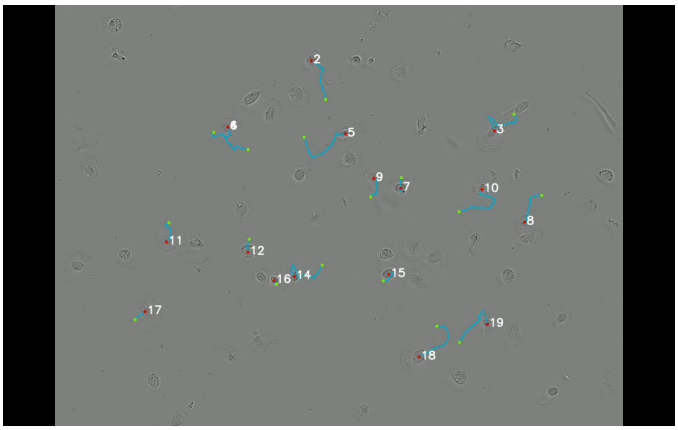


Figure 20. A typical run of the tracker on a sparsely populated sequence.[3]

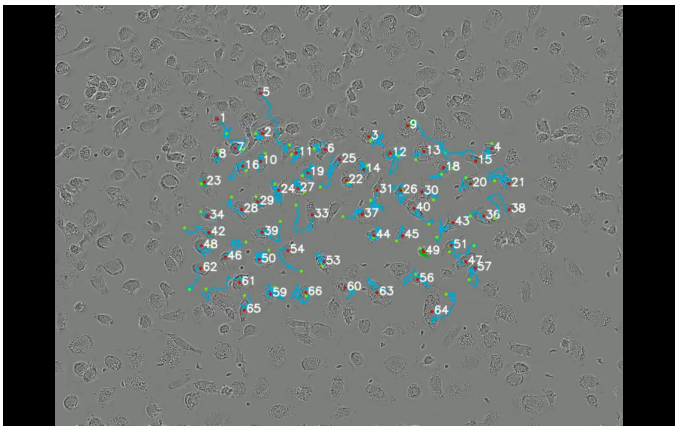


Figure 21. A typical run of the tracker on a sequence with a medium sized population.[5]

7.2 Aggregation

The test results for aggregation are shown below.

3. https://www.youtube.com/watch?v=Mp0_cJPFsaE
5. https://www.youtube.com/watch?v=EFFP3o6CZz0
7. https://www.youtube.com/watch?v=o76drIOZnTA

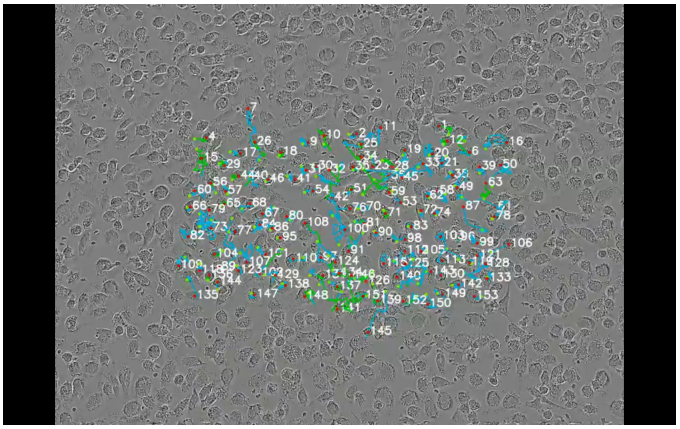


Figure 22. A typical run of the tracker on a densely populated sequence.[7]

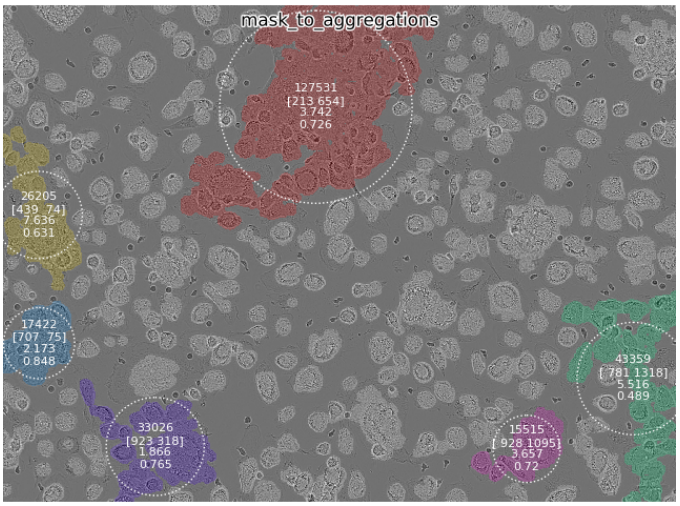


Figure 23. Output image for visual representation of the found aggregations. The numbers in the circle gives information about the aggregation in the following order: size, position, shape ratio and compactness.

7.2.1 Evaluation of aggregation

Output image of detected aggregations can be found in Figures 23 and 24. Evaluation of the two methods used for detecting aggregates can be found in Table 4. A a correct label in 96% of the images was observed with the mask-based method and 95% with the point based from the given data.

8 Discussion

Overall the project has been a success. In both tracking and aggregation detection has been successful. In the previous section we showed that our tracker is very precise and somewhat robust. We also showed that our aggregation detection produces accurate labels for sequences compared to the ground truth. What is somewhat lacking so far is

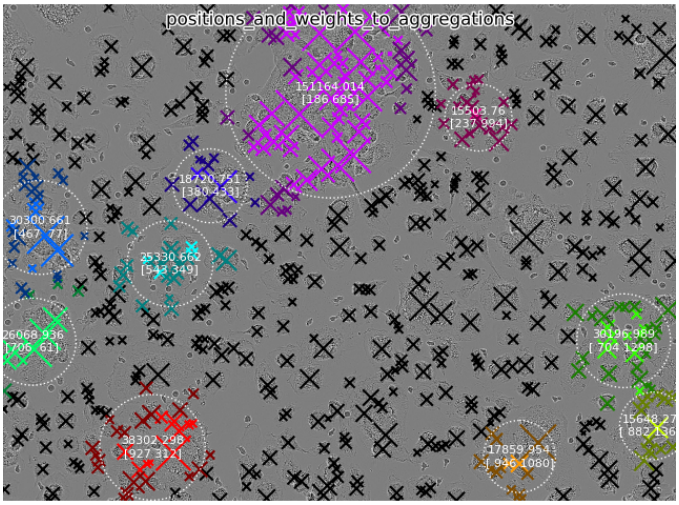


Figure 24. After positions_and_weights_to_aggregations.

Table 4
Ratio of correctly assigned images as Aggregation/No aggregation compared to ground truth.

Experiment	Mask-based	Point-based
w49_2018 (121 frames)	0.992	1.000
w8_2019 (219 frames)	0.982	0.972
w6 (15 frames)	0.933	1.000
w4 (496 frames)	0.944	0.931
Total result (851 frames)	0.960	0.952

further analysis of these results. While we do have the ability to produce several metrics and statistics, we have not conducted a large scale investigation and leave this as future research.

8.1 Tracking

As can be seen in the previous section our tracking works quite well even in challenging sequences, but there are still difficulties. As discussed earlier, cells look very similar to each other, and in some cases even change shape drastically between frames. This causes issues with robustness as the Lucas-Kanade tracker assumes temporal coherence and hence tracking may sometimes completely fail for certain cells.

Cells sometimes create clusters and since the cells all have very similar appearances, it is very hard to distinguish and next to impossible to track the correct cell. Also, since this tool uses images produced by IncuCyte and is used to analyze real crucial data for a scientific purpose, there are some restrictions. You can not make experiments with too few cells since they would then behave differently than they would under normal circumstances.

The given data also have a somewhat bad time resolution on the image sequences, where the best sequences had a time resolution of one image per 30 min. This is also a factor that we can not change without ruining the data, since the cells will behave differently or even die due to the high amount of radiation from the camera. We believe that there is potential for further development of the tracker, and hope future improvements will lead to even better results.

It was observed that the Lucas-Kanade tracker performed the best on lower density sequences while the assignment tracker performed better on higher density sequences. However we were unable to create ground truth data for the high density sequences simply due to the difficulty for humans to accurately label them. This means that most of our quantitative results will show bias to the performance of the Lucas-Kanade tracker. However we still believe that the assignment tracker is an interesting approach, based on promising results from target dense sequences, but more investigation is needed to determine its accuracy.

8.2 Aggregation

As seen in Table 4, both methods performs very well. It seems like the point-based method performs better on some experiments and worse on others compared to the mask-based. Our ground truth shows that some experiments are easier and that explains why we have ratio 1:1 on two experiments (in point-based method). In experiment w49_2018, almost all images are labeled as 'yes' in ground truth, and here the aggregations are large and clearly visible, in which the point based method seems to perform better. In w6 we only have 15 frames which makes it easier to get a higher score. Overall, looking at all the frames in total the mask-based method is better and can correctly label 96% of all images in the data, but the point-based method might be preferred when aggregations are large and clearly visible.

Since the images from different wells in the plates differ and sometimes light and contrast will change over time, the most time-consuming task was to manually tune parameters and finding the best value that fits all images. The evaluation was only made for large aggregations, but smaller aggregations might also be of interest for the researchers, which is why the performance of our solution should be further evaluated. Sources of uncertainty will be discussed in Section 8.3.

8.3 Sources of uncertainty

The created ground truth to evaluate aggregation is based on the given data. Of these images roughly 90% contained aggregations since a significant amount of those experiments had a large density of cells. Due to the more populated density aggregations will still be found even if there are no clumps but the cells are close enough to be considered as such. Also the manual labeling was somewhat challenging when considering an image as aggregation or not. To make a better ground truth another experiment with fewer cells should be included.

9 Future Work

While we have greatly improved the automation of the analysis we still have several areas of improvement which are of interest to future research:

1) Tracking in densely populated sequences is still somewhat unstable, especially when the tracked cells move quickly. We believe this is caused by the tracker attaching to other nearby cells, which 'blocks' the tracking.
2) Further statistics derived from the cell trace information of the cells. We believe that the information we have extracted is not complete. For example detecting cell to cell spread of bacteria should be feasible to predict from just the combined cell traces of all the cells. We also believe it may be possible to combine trace information with frame information to record additional temporal information about individual cell behaviour.
3) Combining the forward pass of the data with a backward pass for improved tracking and error detection. Since we always have access to the full sequences it should be possible to improve results by taking into consideration the inverse movement of the cells.
4) Implement a window that matches the specific cell better when looking for infection and/or death. Instead of what we got now which is a fixed windows size
5) Detecting different forms of cell aggregation and categorizing them. Right now we provide simple statistics for the detected cell aggregates, but it should be possible to provide more detailed descriptions using a priori information.

6) At the moment we are able to register aggregate statistics over time, e.g. sum of aggregate sizes as in Figure 25. However, no method to evaluate the correctness has been developed.
7) Implement the third method in aggregation: a point based method, utilizing the trained U-Net model mentioned in the Section 5.2.3.

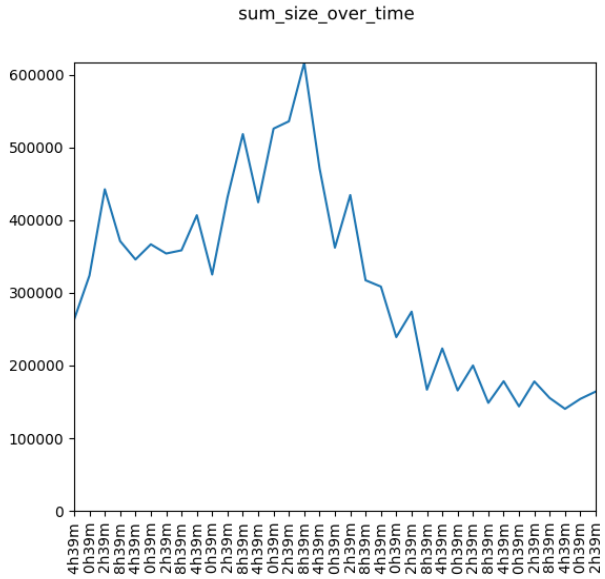


Figure 25. The sum of aggregate sizes over time.

10 Conclusion

In this report we have demonstrated two different methods for automated analysis of microscopy imagery which we have shown perform well.

We have implemented two robust multi target tracking algorithms. One, LK, which is very good in sparsely populated sequences and another, Assignment, which is able to retain identities even in target-dense situations. We use this tracking to extend our analysis in several ways, e.g. calculating velocities, detecting infection and cell death, and a multitude of derivative statistics computed from this information.

We have also implemented automatic detection and description of cell aggregation with two approaches and quantitatively evaluated the methods. One approach was a mask-based method using morphological operations and the other was a point-based method with a clustering algorithm. We conclude that the mask-based performs better than the point-based method on more difficult data while the point-based method is preferred when the aggregations are large and clearly visible. The framework developed can easily be extended and adapted for studying aggregations in other experiments.

Acknowledgement

We would like to thank Maria Lerm, Clara Braian, Jyotirmoy Das and their research group at Linköping University Hospital as well as our supervisor Emanuel Sanchez Aimar at the Computer Vision Laboratory at Linköping University for their advice and guidance throughout this project. We also want to thank our examiner Mårten Wadenbäck for valuable comments and suggestions.

References

Ardila, D., Kiraly, A., Bharadwaj, S., Choi, B., Reicher, J., Peng, L., Tse, D., Etemadi, M., Ye, W., Corrado, G., Naidich, D. and Shetty, S. (2019). 'End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography'. In: *Nature Medicine* 25, p. 1. DOI: 10.1038/s41591-019-0447-x.

argparse — Parser for command-line options, arguments and sub-commands — Python documentation (2019). URL: https://docs.python.org/3/library/argparse.html#module-argparse (visited on 05/12/2019).

Bashir, F. and Porikli, F. (2006). 'Performance Evaluation of Object Detection and Tracking Systems'. In: *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*,

Dinov, I. D. (2016). 'Volume and Value of Big Healthcare Data.' In: *Journal of medical statistics and informatics* 4. DOI: 10.7243/2053-7662-4-3.

Dorra Riahi, G.-A. B. (2016). 'Online multi-object tracking by detection based on generative appearance models'. In: Volume 152, November 2016, Pages 88-102. DOI: https://doi.org/10.1016/j.cviu.2016.07.012.

Ellis, E. L. and Delbrück, M. (1939). 'The growth of bacteriophage'. English. In: *Journal of General Physiology* 22.3, pp. 365–384.

Ernst, J. D. (1998). 'Macrophage Receptors for Mycobacterium tuberculosis'. In: *Infection and Immunity* 66.4, pp. 1277–1281. ISSN: 0019-9567.

Ge, J., Wood, D., Weingeist, D., Prasongtanakij, S., Navasumrit, P., Ruchirawat, M. and Engelward, B. (2013). 'Standard Fluorescent Imaging of Live Cells is Highly Genotoxic'. In: *Cytometry. Part A : the journal of the International Society for Analytical Cytology* 83A. DOI: 10.1002/cyto.a.22291.

GNOME Virtual file system (GVfs) (2019). URL: https://wiki.gnome.org/Projects/gvfs (visited on 12/12/2019).

Heart rate notifications on your Apple Watch (2019). URL: https://support.apple.com/en-us/HT208931 (visited on 22/11/2019).

IncuCyte® S3 Live-Cell Analysis System (2019). URL: https://www.essenbioscience.com/en/products/incucyte/incucyte-s3/ (visited on 25/11/2019).

Johansson, B. (2007). *Derivation of the Lucas-Kanade Tracker*. URL: https://www.cvl.isy.liu.se/education/undergraduate/tsbb15/reading-material/LKderivation.pdf (visited on 26/11/2019).

Kraus, O. Z., Grys, B. T., Ba, J., Chong, Y., Frey, B. J., Boone, C. and Andrews, B. J. (2017). 'Automated analysis of high-content microscopy data with deep learning'. In: *Molecular Systems Biology* 13.4, p. 924. DOI: 10.15252/msb.20177551.

Kuehl, C. J., Dragoi, A.-M., Talman, A. and Agaisse, H. (2015). 'Bacterial spread from cell to cell: beyond actin-based motility'. In: *Trends in Microbiology* 23.9, pp. 558–566. ISSN: 0966-842X.

Kuhn, H. W. (1955). 'The Hungarian method for the assignment problem'. In: *Naval Research Logistics Quarterly* 2.1-2, pp. 83–97. DOI: 10.1002/nav.3800020109.

Lucas, B. D. and Kanade, T. (1981). 'An Iterative Image Registration Technique with an Application to Stereo Vision'. In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*. IJCAI'81. Vancouver, BC, Canada: Morgan Kaufmann Publishers Inc., pp. 674–679.

Mahamed, D., Boulle, M., Ganga, Y., Mc Arthur, C., Skroch, S., Oom, L., Catinas, O., Pillay, K., Naicker, M., Rampersad, S., Mathonsi, C., Hunter, J., Sreejit, G., Pym, A. S., Lustig, G. and Sigal, A. (2017). 'Intracellular growth of Mycobacterium tuberculosis after macrophage cell death leads to serial killing of host cells'. English. In: *eLife* 6.

OpenSSH (2019). URL: https://www.openssh.com (visited on 12/12/2019).

Otaka, A., Takahashi, K., Takeda, Y. S., Kambe, Y., Kuwana, Y., Tamada, Y. and Tomita, N. (2014). 'Quantification of Cell Co-Migration Occurrences During Cell Aggregation on Fibroin Substrates'. In: *Tissue Engineering Part C: Methods* 20.8, pp. 671–680. DOI: 10.1089/ten.tec.2013.0344.

Ranjan, A. and Black, M. J. (2016). 'Optical Flow Estimation using a Spatial Pyramid Network'. In: *CoRR* abs/1611.00850.

Raspberry Pi (2019). URL: https://www.raspberrypi.org (visited on 12/12/2019).

Ronneberger, O., Fischer, P. and Brox, T. (2015). 'U-Net: Convolutional Networks for Biomedical Image Segmentation'. In: *CoRR* abs/1505.04597. arXiv: 1505.04597.

Satrughan Kumar, J. S. Y. (2016). 'Video object extraction and its tracking using background subtraction in complex environments'. In: Volume 8. 2016 Sep; Pages 317-322. DOI: https://doi.org/10.1016/j.pisc.2016.04.064.

Sommer, C. and Gerlich, D. W. (2013). 'Machine learning in cell biology – teaching computers to recognize phenotypes'. In: *Journal of Cell Science* 126.24, pp. 5529–5539. ISSN: 0021-9533. DOI: 10.1242/jcs.123604.

SSH File Transfer Protocol (2019). URL: https://www.ssh.com/ssh/sftp (visited on 12/12/2019).

sshfs (2019). URL: https://github.com/libfuse/sshfs (visited on 12/12/2019).

University, L. (2019). *Hur kan vi hitta nya sätt att stoppa tuberkulos?* URL: https://liu.se/medarbetare/marle69 (visited on 22/11/2019).

Zernike, F. (1942). 'Phase contrast, a new method for the microscopic observation of transparent objects'. In: *Physica* 9.7, pp. 686–698. ISSN: 0031-8914. DOI: https://doi.org/10.1016/S0031-8914(42)80035-X.