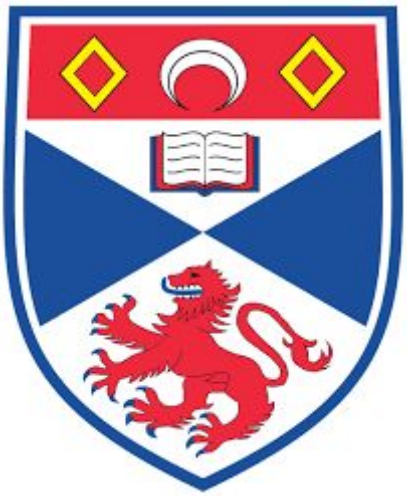
University of St. Andrews

CS4099

Automatic Identification of Cell Motion, Splitting and Death in a 4D Dataset



Author: Peter Goad
Supervisors: Dr. David Harris-Birtill, Dr. Marcus Bischoff

Abstract:

This project aims to expand the scope of software produced by earlier projects within the Computer Science department University of St. Andrews in conjunction with and for the benefit of the Biology Department. This project and the earlier projects it builds on is intended as a research aid for biological research involving microscope images and time lapses. This form of data is of particular interest for studies concerning organ and tissue development, but it is also very time consuming to process manually, which in turn often requires additional research staff to carry out the task and limits the amount of data which can be meaningfully processed. Previous versions of the software automated the process of detecting cells in a set of microscope images, tracking their positions over time and identifying cell deaths in a simplistic form. This project should iterate on these earlier versions, allowing the resultant software to also identify cell splitting events and to register cell deaths with greater accuracy.

Declaration:

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated. "The main text of this project report is 6831 words long, including project specification and plan. In submitting this project report to the University ofSt Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work.

COVID-19 disruption:

This project was subject to disruption due to COVID-19 and the related university shutdown and advice. I had planned to set aside a significant amount of time over the spring break to work on the project: however, I ended up having to plan and book a much earlier flight home, during the first week of the break as opposed to at the start of the second. I also had to plan out what to pack as hold luggage and how, as I did not expect to be returning to St Andrews. This included the PC I would need to use for home/remote work on this practical, which I had to disassemble and pack as hold luggage, and additionally order a replacement monitor and heat sink when I arrived home. After returning home and obtaining these replacement parts, I spent most of the second week of the break setting up remote access to lab computers, so that I could test my code there as well. This took longer than anticipated, as I needed to use Cygwin/X in order to access PyCharm remotely, and this was not directly covered in the systems wiki. Additionally, I was no longer able to meet personally with Dr. Marcus Bischoff, making it more difficult to access School of Biology resources such as the computers

required to run SIMI Biocell. This disruption and additional work requirement meant that I was effectively unable to work on the project for the majority of the spring break, and continued to hamper testing and evaluation going forward, especially as a good part of later development and evaluation overlapped with a marking period for Dr. Bischoff, which also served to slow down communication and development.

Introduction and Objectives:

While microscope images and timelapses are a rich source of data for biological research, said data typically has to be extracted and processed manually, which is a time consuming process and often requires additional research staff. [1] For the same reasons, only a small subset of this data is typically extracted.Therefore, a means of utilising computer vision and image processing techniques to extract this data programmatically would readily address both issues and therefore expedite and increase the comprehensiveness of a large number of biology research projects. Indeed, several implementations of such a system already exist, which served as guidelines for this project, are discussed as such later in the literature review and could potentially be able to serve in the same capacity for future work.

A specific area of study which relies on microscope time lapses, and the primary source of test data for this project, is the study of insectoid metamorphosis, specifically in the genus *drosophilia* ("fruit flies") . Researchers in this area are particularly interested in the position of cells over time, the time and frequency of cell splitting (mitosis) and death events, and the shape of individual cell membranes as they advance or settle into place as part of the adult epidermis [2], and hence a system used in this field of study should be able to provide all of these data types. A system capable of registering these events could also be ported to other areas of study with a similar degree of cell motion, splitting and apoptosis, such as studies of human organ development. [3]

This project aims to build on previous project implementations made in conjunction with Dr. Marcus Bischoff and his work on tracking cell motion, splitting and death. Previous implementations produced a tool which reliably segments and identifies cells as objects, tracks their movement and predicts cell death based on sufficiently long absences in cell visibility, and outputs its results in an SBD file similar to ones produced by manual tracking software. However, the previous implementations also stopped short of detecting cell splitting/mitosis events, and recording or processing any data related to them. Thus, the objectives for this project are as follows:

Primary:

1. Carry out and compile a literature review in order to contextualize the use cases for this system, and to examine the techniques and practices used in other, similar research environments in order to better inform this project's implementation.
2. Extend the provided system so that it is capable of identifying cell splitting events, properly track the resulting daughter cells as new objects and store/output the lineage and generational information of any cells involved in mitosis.

Secondary:

1. Expand the system's object classification process with machine learning techniques, such that it is able to identify cell shapes and membranes in addition to cell centroids.
2. Use this information to augment the system's ability to identify cell splitting and death events by using cell morphology as an additional predictor alongside the present/primary objective implementation.

3. Modify the output file structure such that automatically generated .sbd files can be loaded into SIMI Biocell and examined alongside their source video data.

Software Engineering Process:

In order to make efficient use of my time alongside my work on other coursework submissions, and in order to produce and work towards an effective lower-level definition of the objectives discussed above, I approached this project with an Agile development methodology in which I broke down the stated objectives into smaller tasks and worked iteratively on those tasks and any issues which arose whilst working towards them. I met with my supervisor on a weekly basis to discuss my work, ask for suggestions and evaluate which tasks and goals should be treated as a higher priority for the upcoming week in a manner similar to a scrum meeting.

Literature Review:

Cell biology within context

Cell shape and size fluctuates in a predictable fashion prior to mitosis or apoptosis, the latter being the “programmed” form of cell death which frequently occurs as part of metamorphosis. [4] Hence, automatic classification of these morphological changes in a microscope image or time lapse may help to serve as a predictor of these events (in order to improve accuracy) and more clearly establish the timeframe in which they take place. More specifically, cells undergoing apoptosis contract towards their nuclei (and, if under the effects of luminous dyes or the like, increase in brightness as the dye becomes more concentrated) before rupturing. As this process is usually induced by chemical signals [5], a system that can additionally identify the *start* of an apoptosis event based on this behaviour would be of additional use to studies focusing on this subject area by providing a more comprehensive time frame.

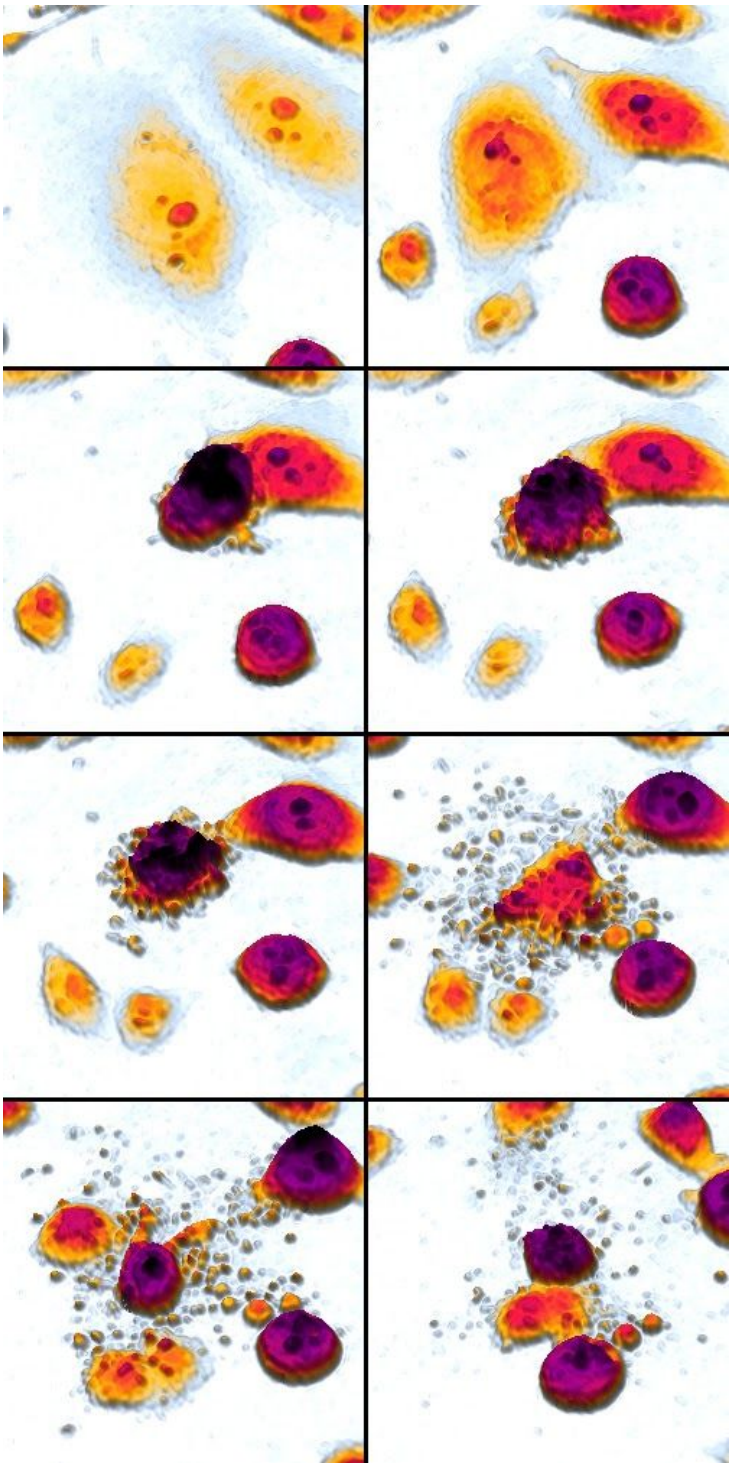


Figure 1: A time lapse showing human prostate cancer cells undergoing apoptosis due to chemotherapy. Note the central cell in the first (top-left) frame which, in the following frames, shrinks, increases in optical density and adopts a more rounded shape before rupturing into many smaller bodies. These morphological behaviors can also be expected in the context of this project's source data sets.[18]

Similarly, mitosis includes several stages with notable effects on cell morphology, particularly the metaphase (wherein a cell's chromosomes are gathered into a flattened "plate" along their centroid) and anaphase (wherein these chromosomes are copied and begin to pull

apart from each other immediately before forming separate cells). Similarly to apoptosis, these stages cause the cell membrane to decrease in size and increase in optical density (though to a lesser extent), though mitotic cells eventually adopt a more ovoid shape as opposed to a uniformly rounded one. [6] As this process takes place over a fairly long span of real time (on the order of an hour), individual time lapse frames are likely to capture each stage of mitosis, making it easier to utilise these morphological indicators within this project's code.

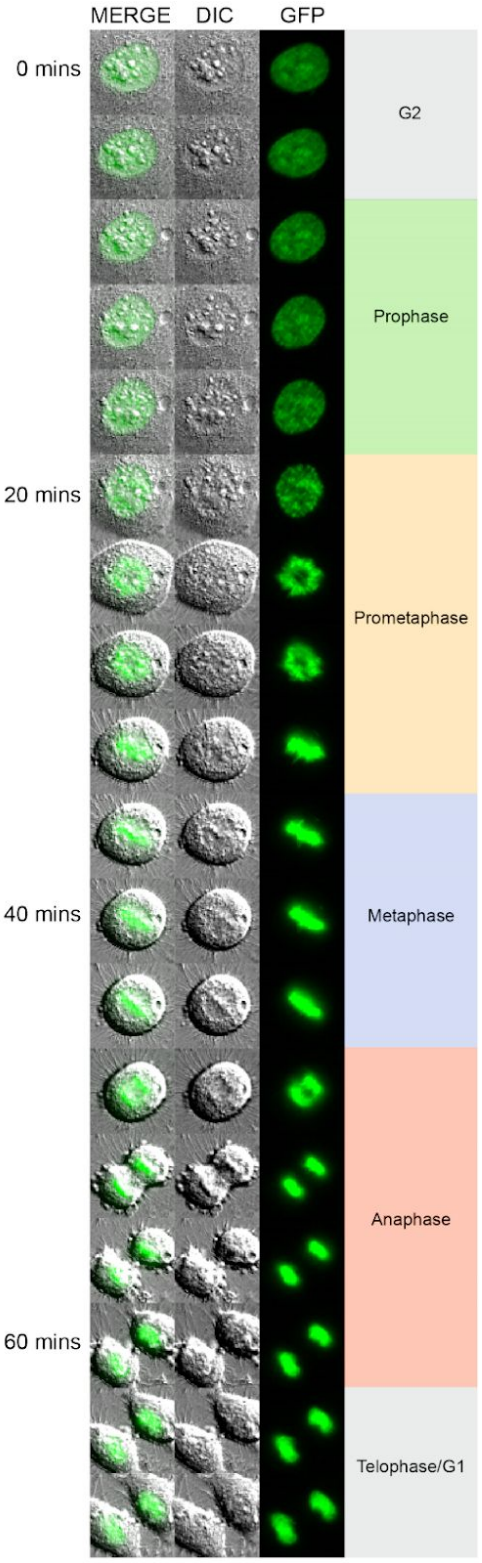


Figure 2: A time lapse outlining the changes in cell membrane (left) and chromosome (right) morphology in an animal cell (specifically the HeLa line of human research cells) as it undergoes mitosis. Note the condensed, rounded shape adopted in the metaphase and the oblong shape exhibited in the first frame of the anaphase. These morphological patterns are common to most animal cells, including the Drosophilia cells this project focuses on. [19]

Cell motion can also be associated with changes in cell shape. During *drosophilia*'s metamorphosis, epithelial cells will move under their own power into their appropriate positions in the tissue layer, and while doing so will adopt a more pointed "D" shape similar to advancing amoebas, or other amorphous single celled organisms. [7] A tracking system which can identify this advancing behaviour could use it as a predictor of that cell's direction of movement, both within the context of a single frame and in addition to positional data from future frames.

Motion tracking for Mitosis Detection:

One advantage of working with a four-dimensional dataset in this context is that cell trajectories and velocities can be extracted from their positional data over time. This additional data could be used in conjunction with assumptions about the problem space to provide a rough predictor of cell mitosis; supposing that we can assume or establish a certain minimum distance between cells in a frame, we can deduce that any cells predicted to be closer to each other in a prior frame, based on a regression of their velocity, are likely to have been involved in a mitosis event. Some implementations along these lines leverage additional probabilistic and motion-based techniques such as Markov dependency chains or vector fields [8] alongside this general idea. Others produce a more refined internal representation of a three dimensional space than a collation of stacked two dimensional images by collating this data into a three-dimensional, voxel-based image (voxels being analogous to two-dimensional pixels) and from there attempting to identify the precise location of objects or regions of interest compared to the boundaries of the voxels. This process is referred to as skeletonisation [9], and although it is usually applied to create precise models of large, complex structures, the cited paper does discuss some principles which may be applicable to this problem space, such as: the explicit collation of data into a three-dimensional structure; the use of distance fields; and the degree of precision offered by implementing an evaluation system that operates within individual voxels as opposed to treating voxels as atomic

Machine learning and object classification

General Techniques:

One of the most popular machine learning models is the neural network, which has been successfully applied to numerous object classification problems in research. A neural network is comprised of any number of individual computational units known as neurons (styled after the biological neuron); Each individual neuron accepts one or more input values, applies some internal function involving these outputs, and passes its output to one or more other neurons, with one (set) of neurons eventually passing output for the system altogether. These systems can then "learn" by being provided with some metric for correct output, either in the form of numerical test or validation data (for numerical/"regression" problems) or pre-generated input/output pairs (in the case of categorical/"classification" problems, including most machine vision problems), which allows the system to calculate the error

between its output and the correct output, and adjust its own parameters based on this error. For neural networks, this typically occurs by altering the weights assigned to each neuron connection, which corresponds to a modifier affecting the impact of that connection's value on the function when it is passed to a successive neuron. [10] The mathematical basis of these neurons usually involves matrix multiplication, though alternatives do exist; convolutional neural networks, for example, apply mathematical convolution as part of at least some of their layers, allowing output in their intermediate layers to more accurately reflect the combination of smaller, less complex input patterns into a singular output. [11]. This makes them particularly suitable for object classification tasks, with the majority of recent research I found on this topic making use of CNNs in some way.

This research also implies that a large number of more advanced object recognition systems are based on a combination of a CNN with another machine learning/object classification model, wherein the second model affects or informs the results of the CNN itself. One example is the Mask Receptive CNN [12], which includes an additional neural network implementation, a "fully connected network" or FCN attached to intermediate layers of the base CNN. A fully connected network is a neural network wherein all neurons in any given layer are connected to all neurons in the previous layer, making them more generalised, but also less able to exploit known information about a problem and so more expensive to train. These issues, however, do not arise in the context of this paper, as the FCNs used are small and applied to small-scale tasks; namely, predicting a segmentation mask for a single region of interest in parallel with the CNN, which attempts to classify that entire region as containing an object and then establish the smallest rectangular section which contains the entire object, based on the results of all regions. This allows for a segmentation mask to be produced as part of the classification process with very little extra computational overhead. This paper and its approach are of particular interest for this problem space, as refined object location predictions and pixelwise segmentation of said objects would be extremely helpful in the processing of microscopy data featuring many densely packed cells.

Another example I encountered involved using a CNN in conjunction with an extreme learning machine, or ELM [13] . An extreme learning machine is comprised of a neural network with a single, non-recurrent hidden layer (meaning that it consists only of this layer and its input/output layers) that does not use a traditional back-propagation learning method, and instead randomly generates the input weights and neuron-internal function variables, while output weights from the hidden layer are allocated by treating the connection values from the hidden layer as a matrix, and evaluating the inverse of that matrix in such a way as to minimise error across these values when they are applied as multipliers to each neuron's connection to the final output neuron. This results in an ELM having a much faster learning process than backpropagation learning systems, and one which is not subject to many pathological cases or parameter management issues that often apply to them. The system in the cited paper uses three of these networks, two to extract and represent features in terms of image depth and RGB content, and a third to collate the features produced by these networks into a final classification output. The ELM model offers fast training and generalisability, which are both ideal qualities for a system intended to work with multiple different types or "styles"of microscopy data (for example, whether or not the cells are treated with dye prior to imaging), though I am less confident that the multi-modal

component would be portable to microscopy processing, as the majority of images I have observed are either monochrome or have limited colour variation.

Applications within Microscopy:

Biology research papers have, for some time, acknowledged the utility and importance of automated cell tracking systems, or at least systems which can speed up the process of manual cell tracking. [14]. Of particular note in this discussion is how it highlights several issues with basic, existing segmentation techniques: how uninformed segmentation techniques can produce inaccurate data, which leads to increased processing time due to the manual corrections required; limited capacity to carry out more complex, yet frequently required, segmentation processes, often requiring specific input data or a much longer processing period to do so; and the difficulty of sharing and generalising techniques used in one lab to other labs and research groups. This paper then goes on to discuss recent applications of machine learning and Convolutional Neural Network techniques to biological data, and how these applications can address the main concerns with existing cell processing systems discussed earlier in the paper.

Object recognition in this context presents an added focus on identifying multiple distinct objects within a single image, and creates an additional requirement for pixelwise segmentation due to the large number and close proximity of cell objects in a typical frame. Indeed, the Mask Receptive CNN system described earlier in the paper has been used as part of a cell recognition implementation [15]. The use and discussion of Mask R-CNN confirms and re-iterates my earlier points about its ability to produce accurate, pixelwise object masks, and the usefulness of these properties in the context of microscopy analysis. This implementation additionally features direct, side-by-side comparison between images and tracking masks, allowing them to be readily verified by a human operator without requiring them to manually create said tracking masks; this property could be used to verify and address issues in the training results and process for a similar implementation.

System Requirements

Software used:

Note that these software requirements are predominantly inherited from the provided implementation. I saw no need to replace them and hence I will provide an overview of their relevance:

Python 3.7:

Python is a high-level programming language, which was used to implement the provided code base due to its flexibility and access to a large number of external libraries for image processing, machine learning and other lower-level functionality. Earlier versions of this

project appear to have been made with the depreciated Python 2.X in mind, but I encountered no issues in porting the project to Python 3.7. This upgrade additionally renders the multiprocessing and CSV libraries used in earlier projects unnecessary, as Python 3.X natively supports these features.

PyCharm:

PyCharm is an integrated development environment designed specifically for Python, which greatly simplifies the process of finding and installing external libraries and provides access to a debugger for testing and development purposes. It is currently the preferred means of setting up and running the system due to the large number of dependent libraries to be installed.

Hardware and Operating System:

In earlier stages of the project, the system was run on a Fedora implementation of Linux. However, I left St. Andrews due to COVID-19 partway through the project, and at this stage it became more convenient to run the system on my personal computer, which uses Windows 10. Despite some minor unexpected behaviour on Windows, (the program opens multiple GUI windows when it begins multithreading, which, excluding the original window, can all be closed without disrupting the program), the program runs successfully on both systems, and therefore I can reasonably conclude that the system will work on any operating system which supports Python. Note that, due to the large number of image segmentation and computation tasks involved in running the program, a computer with a reasonably powerful GPU and CPU is recommended.

TKinter:

TKinter is a standard, lightweight graphical user interface package for Python, used to more easily display and allow selection of segmentation options, program execution etc. It supports Windows, Linux and Mac OS operating systems, and is typically included by default in Python installations for those operating systems.

SIMI Biocell:

SIMI is the main manual cell tracking system used to support this practical and provide manual tracking reference data for it. It allows for side-by-side comparison of a microscopy time lapse and the cell tracking data associated with it, along with allowing said tracking data to be added manually by clicking on locations in the time lapse frame. Though not explicitly required to run this project's artifact, I considered it worth mentioning due to its intended role in providing and evaluating the data used for development and testing, and due to the additional steps taken to make the artifact's output compatible with SIMI. It is also worth noting that SIMI is paid software and runs only on a small selection of operating systems, such as Windows 7 or 32-bit Windows 10, making it difficult to access.

Libraries:

PIL and CV2:

PIL, or Python Imaging Library, is an image processing and manipulation library that allows for the loading of image files into an efficient internal representation. It is used in conjunction with CV2, a similar library which is focused on functions and techniques related to computer vision, in order to load the .tif files used to store images from the microscope time lapse at specific points in time and at specific z-levels. Note that these libraries were installed in the project's virtual environment using forks (Pillow and opencv-python respectively).

SciPy:

SciPy is a library which provides access to a number of mathematical functions and techniques predominantly used in scientific research or other science-related program applications. In this project, it implements one of the prerequisite stages to performing watershed segmentation on cell input images (that being calculating the Euclidean distance over the whole image).

Numpy:

Numpy is a commonly-used Python library which provides access to more advanced mathematical functions and concepts, including multidimensional arrays and functions required to manipulate them. It is used as part of the existing implementation to collate multiple z-levels in the same time step into a singular data structure, and to provide the mathematical mask used in segmentation.

Scikit-Image:

Scikit-Image is a robust image processing library used to provide the lower-level image segmentation, thresholding and filtering functions used as part of the provided implementation.

Matplotlib and Pandas:

These libraries provide statistical and graphing methods which were used in previous project submissions to carry out statistical analysis over the output of the artifact.I was not able to carry out similar analysis in this project, so I mention these libraries for the sake of completeness.

Ethics:

This project does not source any research data from human or (complex) animal subjects, and is therefore not subject to any special ethical restrictions or considerations. All microscopy data used to develop and test the system was sourced from the University of St Andrews School of Biology and their research into fruit flies, which, being insects, do not present any ethical concerns in regards to animal rights or testing. I have completed a self-assessment form to this effect and provided it alongside this report as an appendix.

Design and Implementation:

Initial Code base overview:

My work on this project consists of extensions to an artifact that was in turn produced and expanded upon by previous projects at the university. In order to provide context for the artifact as a whole, and my additions to it, this section provides a brief overview of the system's design and behaviour prior to the start of this project. This pre-existing system covers image loading, pre-processing and segmentation, locational tracking and basic cell death detection systems. This system also includes a cell object class to store relevant data and convert it to the appropriate .sbd structured output, though some fields (particularly those related to mitosis) are filled with static values.

The base system produces a graphical user interface window for the purpose of selecting the image pre-processing and segmentation techniques to be applied to the input images, along with previewing their effects over a selected .tif file. Results in previous dissertations indicated that a combination of gaussian filtering, adaptive thresholding, binary closing and watershed segmentation produced the best results overall, so these settings were predominantly used for this project. [16]

This system also designates an input folder in the settings.py file and groups all .tif files found in the folder by z-stack, with each individual group being handed off to a parallel computation thread for pre-processing and segmentation. Once the images have been processed, and centroid location data is obtained from the segmentation output, cell movement is tracked using Nearest Neighbour target matching, which consists of calculating the distance between a given cell and each cell in the next frame (based on point distance between centroids) and designating the closest following cell as the initial cell's new position in that frame. [17] Following this, the cells are re-grouped into a single list in the main thread, dead cells are identified by filtering cells which are not tracked for a sufficient length of time after being discovered, and cell information is converted into string form and printed to an output file.

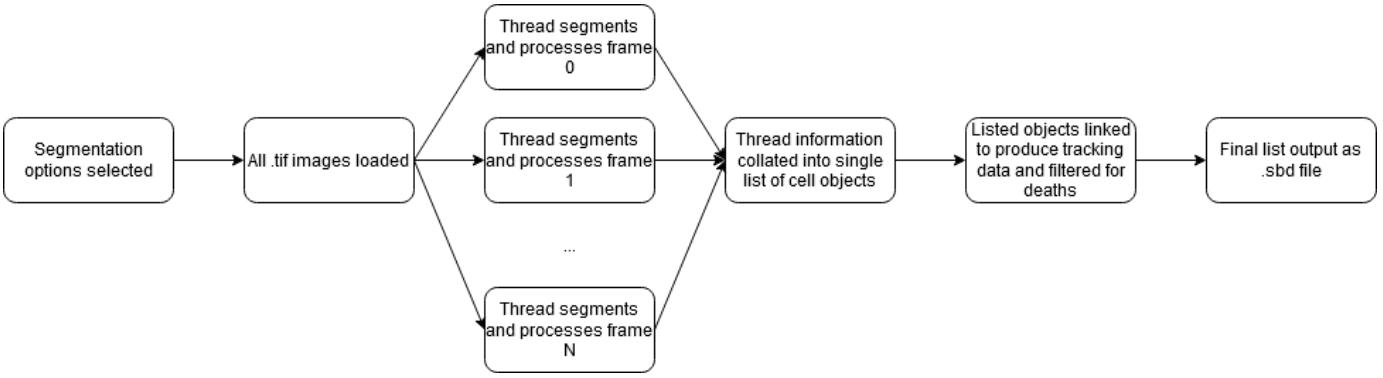


Figure 3: A procedure diagram roughly outlining the functionality and order of operations for the provided code base.

Modified cell data output

My implementation extends the cell object string representation method, and the manner in which it is invoked, in order to encode the information extracted by other new code in line with the SIMI biocell format, which is defined as follows: [embed version with tree traversal?]

```
# Header: # # # # # # #
SIMI*BIOCELL
400
 ---
 # # # # # # # # # # #

SIMI*BIOCELL = magic ID string
400                = file version 4.00
---        = separator


# Cell: # # # # # # # #
<free 3D cells count>
<start frame> <end frame> <x> <y> <level> <comment>     <= *count
---
<start cells count> <start time>
<start generation>                                  <= *count
---
<cells left count> <"right> <active cell left> <"right> <gen.name1>
<gen.time of birth sec.> <g.level> <g.wildtype> <g.color> <g.name2>
<birth frm> <mitosis lvl> <wildtype> <size> <shape> <color> <name>
<coordinates count> <cell comment>
<frame> <x> <y> <level> <size> <shape> <coord.comment>  <= *count
---
# # # # # # # # # # #
```

<start ...> values are not used yet (for later implementation) <color> is a real hexadecimal RGB value (e.g. 00ff00 for green) <size> and <shape> are internal values of BioCell <coordinates> are real pixels

BioCell uses a user-definable list of <wildtypes>. The list has default entries and I believe nobody used the possibility to modify it until now. These are the default entries (see sbiocell.ini and each project file (*.sbc)):

0=A,Aberrant 1=?,Cell lost 2=H,Hypodermis 3=U,Muscle 4=N,Neuron 5=P,Pharynx 6=I,Intestine 7=D,Death 8=M,Mitosis

An Example of Manually Tracked .sbd File Format:

1 1 0 0 ABaaaaaaaa
42000 0 -1 -1 ABalaaaala
23 4 -1 -1 0 33023 ABalaaaala
6
24 170 49 7 -1 -1 -1
25 169 51 7 -1 -1 -1
34 170 60 6 -1 -1 -1
44 168 75 4 -1 -1 -1
45 166 74 4 -1 -1 -1
46 162 80 4 -1 -1 -1

Added information pertains to values in the Cell section, specifically the values for:

- ‘cells left count’ and ‘“right’, which represent the total number of descendants on each side of that cell and are generated by recursively invoking a getter method over that particular cell’s daughters, who in turn invoke it over their daughters and so on.
- ‘gen.name1’, which is retrieved from the cell’s parent, if available.
- ‘gen.name 2’ and ‘name’, which describe the cell itself and are generated from the internal numerical identifiers used to distinguish cells (with ‘l’s or ‘r’s added to indicate daughter cells)
- ‘mitosis level’, describing the overall number of mitosis events which have occured in the cell’s lineage so far. This is effectively encoded into a daughter cell’s name in the form of how many ‘l’s/’r’s have been added.

Additionally, cell generations are padded and ordered in a similar fashion to standard, manually tracked .sbd files. In particular, padding cells are added to the file such that tracked cells without known parents first appear in the tenth generation, and the cell lineage tree is written to the output file in a pre-order traversal pattern. These modifications would ideally allow the file to be loaded into SIMI biocell itself, allowing the output data from the system to be overlaid onto the source microscopy time lapse. However, this could not be achieved during the project’s development cycle.

Mitosis tracking

As a simple means of identifying mitosis prior to implementing object recognition, I decided to instead locate mitosis events in the post-detection cell data by creating a list of all “novel” cells which were not detected in the initial frame of the data set, extrapolating a likely position for those novel cells in the frame prior to their detection based on their movement in later frames (more specifically by acquiring the difference in location between the cell’s initial appearance and its location in the next frame, then assuming that it moves with an equal and opposite velocity for one frame), and attempting to match this cell with another cell in the output data based on this regressed location. This required a modified point distance function, since although the cells’ locations over time are tracked using an extension of the pre-existing Point class, the z-coordinates used in these points correspond to layers in the

overall microscopy video as opposed to coordinates. The pre-existing comparison function hence wrongly interprets them as being in the same order of magnitude as the x and y coordinates and matches cells on extremely distant z-levels from each other.

The overall method is based on the assumption that if any two cell centroids are (predicted to be) closer to each other than a certain multiple the watershed detection threshold already provided in the base code, then it is likely that they resulted from the same cell. At the time of submission, this mitosis threshold was about twice the minimum watershed detection threshold. Since a mitosis event only lasts a few frames and would not cause a significant enough gap in continuity for the cell to cease being tracked, the initial system would in all likelihood continue to track a post-mitotic daughter cell as part of the same cell object. In order to more closely match manual tracking outputs, a new cell instance is created on the detection of a mitosis event and the parent cell's list of locations over time is split so that the new daughter cell can be given the appropriate list of positions, i.e. the parent cell's position at every time point after the cell split.

Though simple in concept, this method does require a large number of comparison operations based on the number of novel cells and the fact that each one would need to be compared against the entire set of detected cells in order for this method to be complete. This in turn means that this section of the code induces a significant time overhead, extending the runtime of the system to upwards of two hours. Though still faster than manual analysis, this does hamper testing and usage across multiple input datasets. Similarly to per-frame segmentation, this process could be sped up through multithreading: however, the code executed upon detecting a mitosis event is non-commutative due to the fact that a new cell is created and that the original paired cell's list of locations over time is split. In other words, two novel cells originating from the "same" cell (or more accurately, one originates from that cell and the other from one of its daughter cells at a later point in time) may encounter a race condition, as both attempt to split from the same cell - which, depending on the order of operations, may have had the appropriate location data split off without informing the younger daughter cell. A multithreaded solution would therefore require some form of synchronisation scheme, and given that the system still runs within a timeframe appropriate to its context, I decided it was not worth implementing such a solution at this time.

Along with these additions to tracking code, the Cell class was expanded to contain and process information related to mitosis events. Cell objects now contain attributes referencing their mother and daughter cells (set to None by default), a "mitosis" function which automatically updates these attributes for the parent cell (on which the function is called) and daughter cells (which are passed as arguments), a function to produce and store the regressed location of a novel cell and previously discussed modifications to the string representation function to display child count, mitosis level, cell/generation names et cetera.

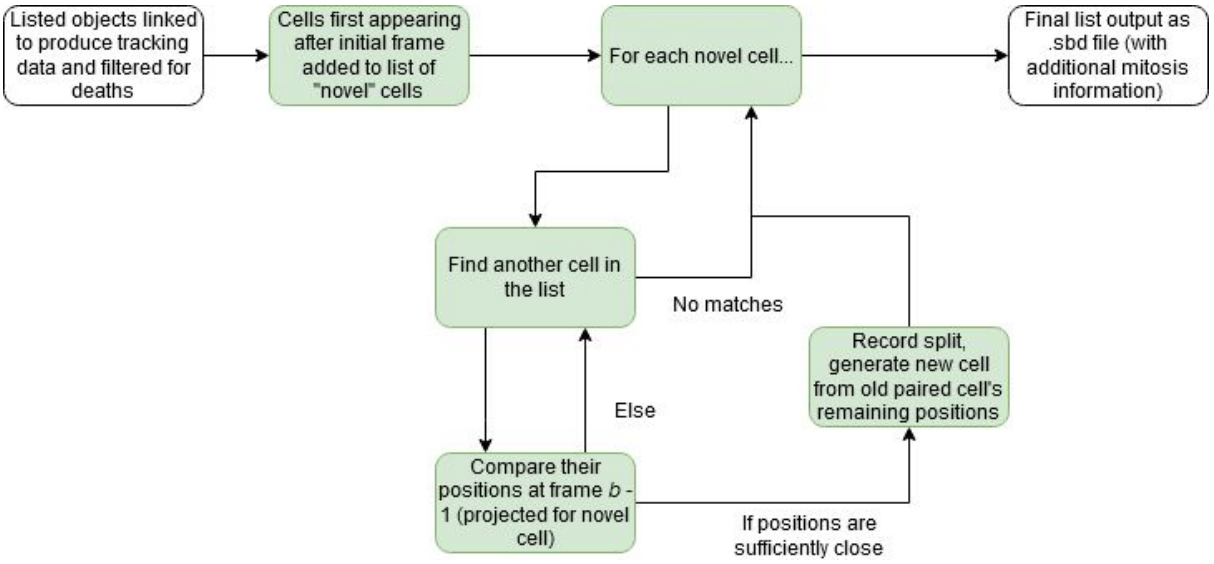


Figure 4: A procedure diagram outlining the mitosis tracking process in my implementation, and where it occurs in relation to the existing program behaviour. Steps added as part of this project are indicated by green-coloured cells.

(Planned) Object classification model:

Given that the intention of this classification model was to accurately reproduce classification categories produced and defined by manual tracking and observation, I thought it best to treat it as a supervised learning task, and therefore as a task which would require a labelled training set. To this end, I again collaborated with Dr. Bischoff and asked him to provide me with highlighted instances of each cell morphological behaviour of interest those being: “normal” cells, pre-mitotic cells, pre-apoptotic cells and migrating cells. Unfortunately, I was unable to obtain this data and begin development in time, and hence I will briefly discuss what I had planned for this implementation and its design.These behavioural types would then form the labels for my classifier. I planned to obtain ten initial examples of each label and then create copies of each using a system of standard image transformations (i.e. rotations, resizing?, horizontal/vertical flips and combinations thereof) in order to effect a system of data augmentation. Since I would be working with a relatively small dataset even with this data augmentation pass, I decided it would be best to make use of a cross validation approach to training and evaluating the model, in order to avoid reducing the size of the training data any further by splitting off a specific validation set.

Results analysis:

In addition to statistical comparison between my output data and a manual tracking “baseline” set out output data, I decided that my project would benefit from manual evaluation of its output data compared to the input video file. I came to this conclusion due to a number of factors: firstly, trying to identify specific, corresponding cells in both datasets may have been difficult, as they would not share a common identity variable and because

there may be significant mismatch between their centroid positions as identified by the program and by the producer of the manual file; additionally, as the “basic” cell split detection method does not directly interpret image data or work from any training set, the only way to directly evaluate its accuracy in comparison to the input data would be to view them side by side, as in SIMI Biocell. These factors motivated the extended SIMI integration objective in my final artifact, and in the development stages of the project, I worked with Dr. Bischoff and provided him with my system’s output data in an attempt to ensure that he would be able to provide me with an evaluation using SIMI.

However, due to project disruption discussed earlier, I was unable to completely implement this system. Therefore, in order to approximate this kind of analysis, I decided instead to modify the statistical evaluation functions provided in the base code to focus specifically on cells involved in mitosis in both files. These functions provide average and root mean squared error values between manual cells and their closest equivalents in the automatic file (though due to the much smaller number of mitotic cells reported in the automated tracking file, I decided to modify the error evaluation method in order to select from automated tracking cells for comparison with manual ones instead of vice versa). Ideally, if cell splits identified by the automated system do not correspond to actual cell splits, they will exhibit an uncharacteristically high RMS error compared to the average over all automated cells.

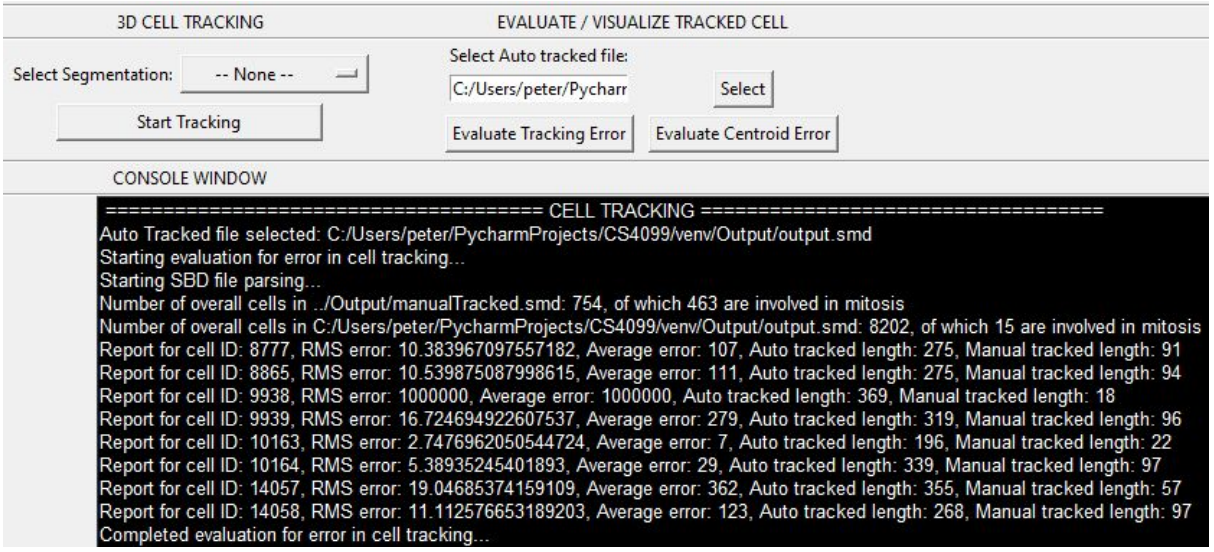


Figure 5: Graphical output for the program’s error evaluation for mitotic cells, which calculates the root mean squared error between each cell in the automated output file and its closest equivalent in a reference manual tracking file. [convert to table?]

This output indicates that the mitotic cells identified by the automated tracking system exhibit a mean RMS of around 10.85 pixels, with a standard deviation of around 5.31. Abishek calculates that the diameter of a cell, based on the upper limit of the code’s cell area filter, is around 7.97 pixels, indicating that the average error is within 1-1.5 cell widths. This mean value was still exhibited by some cells in Abishek’s results for the same combination of filtering and thresholding techniques, but is nonetheless above the mean for his overall results. A few cells exhibit much lower values of 2-5 pixels, which is much closer to the average RMS error for Abishek’s results, while some others exhibit significantly higher

values of 16-19. (The reported RMS value of is 1000000 is, as best I can tell, a mathematical error resulting from the mismatch in automatic and manual tracking length for that cell, though this is in and of itself probably indicative of an error in the detection code. It has been omitted from the calculations above by way of Winsorization.) These results indicate to me that, though the mitosis detection code I have added does identify non-spurious mitosis events and does so with comparable accuracy to Abishek's tracking of regular cells, it is still subject to a significant degree of inaccuracy, as an ideal system would not exhibit error in excess of one cell width. Without the manual evaluation discussed earlier, however, I am unsure as to how to reduce this inaccuracy.

Additionally, I also examined the raw number of mitotic cells compared to the number of cells overall identified by each tracking method, as a measure of the automated system's reliability. The proportion of mitotic cells in the manually identified file, however, may not be representative of the number of mitotic cells overall; due to the time and labour cost of performing manual tracking and the infeasibility of tracking every cell in a dataset manually, it is likely that any manual evaluation will exhibit bias towards "interesting" cells, such as those cells undergoing mitosis, so as to ensure as much useful data for the purposes of their biological study is extracted.

This method of analysis showed that of the 754 (non-padding) cells present in the manual tracking file, 463 are involved in mitosis. By contrast, of the 8202 cells in the automated output file (which only examined the first 100 frames of the video, compared to the 150 covered by the manual file), only 15 are involved in mitosis. Whether or not the manual file is significantly biased, the difference in the raw number of mitotic cells identified between the two files indicates that the base mitosis tracking code is not a sufficiently reliable indicator of cell splits on its own. Likewise, this low number of data points made it difficult to carry out further statistical analysis.

The difference in cell tracking length visible in the evaluation output is likely not worth examining directly, as Marcus pointed out to me that most manually tracked cells do not have their positions reported at every frame they are visible, but rather at points or events of interest, in contrast to the automated system, which *does* log their position at every frame.

Evaluation:

Primary Objective 1 - Literature Review:

The literature review for this project achieved its stated goal of providing an overall explanation and exploration of the problem space, its features and existing solutions or approaches to the problem or to similar problems. In particular, this meant and examination of a number of different cell tracking and machine learning systems discussed in research in recent years. I am also confident that it provides a satisfactory explanation of said approaches and problem space for anyone reading this report who is unfamiliar with either aspect of the project.

Primary Objective 2 - Cell Split Identification:

The final artifact implements a satisfactory basic cell split detection system based on back-tracking and intersection of cell trajectories. I have iterated on and evaluated this system to confirm that it is capable of identifying non-spurious cell split events. However, my results indicate that it is still a very unsatisfactory implementation. I updated the cell to string representation function and the cell object's internal storage in order to ensure all information associated with cell splitting - generations and cell lineage, mitosis level, etc. - are encoded in the output file.

Secondary Objective 1 - ML implementation / Secondary Objective 2 - ML integration:

Due to aforementioned project disruption, I was unable to implement these project objectives.

Secondary Objective 3 - SIMI integration:

In addition to the information on cell splits added as part of the first primary objective, I also modified the output file's structure to more closely match that of regular SIMI files in the hopes it could examined by a manual operator alongside the input data, and therefore provide a source of manual evaluation for the mitosis tracking system implemented and discussed above.

Conclusion:

This project aimed to expand the scope of previous cell tracking implementations to include a wider array of cell properties and events in its output. This additional data includes cell mitosis events, the ability to identify cell membranes and their shapes, and associating fluctuations in membrane shape with events of interest such as mitosis or death. As these events and properties are often of interest to biology researchers, these improvements would ideally make the artifact more useful in its intended role as an aid to microscopy-reliant biology research.

The system implements a functional mitosis detection system based on back-tracking and intersecting cell trajectories over time. Any additional cell data/properties related to mitosis have been added to the cell data output system, which has also been modified in order to produce a result more closely resembling a manually tracked output file. It was planned that these systems would be accompanied by a machine learning model which would identify and classify cell membranes at the tracking stage, and incorporate its output into predictions of cell behaviour and of events of interest, but this system did not exit the research and design phase during the course of the project.

The primary objective of mitosis detection was achieved, but I am not entirely satisfied with its performance. The current system appears to be inaccurate and unreliable, and as I was unable to have its output manually evaluated prior to submission, I am unsure what exactly caused these inaccuracies. My literature research also indicated that tracking implementations in a similar vein to my own (i.e. based on cell position and back-tracking) made use of more advanced probabilistic techniques, and I was not confident that I would be able to integrate these with the basic system without significantly overhauling it at a late stage in development. The secondary objectives for the project were, for the most part, not achieved, as even in spite of numerous attempts to bring the system's output file in line with the SIMI biocell format and with manually tracked data, attempting to load it into SIMI biocell produced unexplained errors. While the machine learning implementation is not required for the system to achieve its stated goals, it would potentially serve to greatly increase the accuracy of its event prediction components. While I am overall dissatisfied with the final state of this project, I conclude that it is at the very least functional and achieves its primary objectives as stated earlier in this report.

References:

1. Cordelières, Fabrice P., et al. "Automated cell tracking and analysis in phase-contrast videos (iTrack4U): development of Java software based on combined mean-shift processes." *PloS one* 8.11 (2013).
2. Bischoff, Marcus, and Zoltán Cseresnyés. "Cell rearrangements, cell divisions and cell death in a migrating epithelial sheet in the abdomen of Drosophila." *Development* 136.14 (2009): 2403-2411.
3. Kan, Andrey, et al. "Automated and semi-automated cell tracking: addressing portability challenges." *Journal of Microscopy* 244.2 (2011): 194-213.
4. Nagata, Shigekazu. "Fas-mediated apoptosis." *Mechanisms of Lymphocyte Activation and Immune Regulation VI*. Springer, Boston, MA, 1996. 119-124.
5. McConkey, David J., and Sten Orrenius. "Signal transduction pathways in apoptosis." *Stem Cells* 14.6 (1996): 619-631.
6. Shkolyar, Anat, et al. "Automatic detection of cell divisions (mitosis) in live-imaging microscopy images using convolutional neural networks." *2015 37th annual international conference of the IEEE engineering in medicine and biology society (EMBC)*. IEEE, 2015.
7. Maeda, Yusuke T., et al. "Ordered patterns of cell shape and orientational correlation during spontaneous cell migration." *PloS one* 3.11 (2008).
8. Boukari, Fatima, and Sokratis Makrogiannis. "Automated Cell Tracking using Motion Prediction-based Matching and Event Handling." *IEEE/ACM transactions on computational biology and bioinformatics* (2018).
9. Van Uitert, Robert, and Ingmar Bitter. "Subvoxel precise skeletons of volumetric data based on fast marching methods." *Medical physics* 34.2 (2007): 627-638.
10. Hecht-Nielsen, Robert. "Theory of the backpropagation neural network." *Neural networks for perception*. Academic Press, 1992. 65-93.

11. Liang, Ming, and Xiaolin Hu. "Recurrent convolutional neural network for object recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.
12. He, Kaiming, et al. "Mask r-cnn." *Proceedings of the IEEE international conference on computer vision*. 2017.
13. Liu, Huaping, et al. "Multi-modal local receptive field extreme learning machine for object recognition." *Neurocomputing* 277 (2018): 4-11.
14. Van Valen, David A., et al. "Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments." *PLoS computational biology* 12.11 (2016).
15. Tsai, Hsieh-Fu, et al. "Usiigaci: Instance-aware cell tracking in stain-free phase contrast microscopy enabled by machine learning." *SoftwareX* 9 (2019): 230-237.
16. Automatic Identification of Cell Motion, and Cell Death in a 4D Dataset - CS5099, Abishek Kargawal, August 2019
17. Automatic Identification of Cell Motion, Splitting, and Death in a 4D Dataset - CS4099, Gareth Cairns, April 2019
18. “Apoptosis DU145 cells mosaic“ , https://commons.wikimedia.org/wiki/File:Apoptosis_DU145_cells_mosaic.jpg, last accessed 27 April 2020
19. “CellShape vs M-Phase”, https://commons.wikimedia.org/wiki/File:CellShape_vs_M-phase.png, last accessed 27 April 2020

Appendix:

UNIVERSITY OF ST ANDREWS
TEACHING AND RESEARCH ETHICS COMMITTEE (UTREC)
SCHOOL OF COMPUTER SCIENCE
PRELIMINARY ETHICS SELF-ASSESSMENT FORM

This Preliminary Ethics Self-Assessment Form is to be conducted by the researcher, and completed in conjunction with the Guidelines for Ethical Research Practice. All staff and students of the School of Computer Science must complete it prior to commencing research.

This Form will act as a formal record of your ethical considerations.
Tick one box

- ☐ **Staff Project**
- ☐ **Postgraduate Project**
- ☒ **Undergraduate Project**

Title of project

Automatic identification of cell motion, splitting, and death in a 4D dataset

Name of researcher(s)

Peter Goad

Name of supervisor (for student research)

David Harris-Birtill, Marcus Bischoff

OVERALL ASSESSMENT (to be signed after questions, overleaf, have been completed)

Self audit has been conducted **YES** ☒ **NO** ☐

There are no ethical issues raised by this project

Signature Student or Researcher

P. Goad

Print Name

Peter Goad

Date

26/9/~~2018~~2019

Signature Lead Researcher or Supervisor

Print Name

DR DAVID HARRIS-BIRTILL

Date

26/9/2019

This form must be date stamped and held in the files of the Lead Researcher or Supervisor. If fieldwork is required, a copy must also be lodged with appropriate Risk Assessment forms. The School Ethics Committee will be responsible for monitoring assessments.

Computer Science Preliminary Ethics Self-Assessment Form

Research with human subjects

Does your research involve human subjects or have potential adverse consequences for human welfare and wellbeing?

YES ☐ **NO** ☒

If YES, full ethics review required
For example:
Will you be surveying, observing or interviewing human subjects?
Will you be analysing secondary data that could significantly affect human subjects?
Does your research have the potential to have a significant negative effect on people in the study area?

Potential physical or psychological harm, discomfort or stress

Are there any foreseeable risks to the researcher, or to any participants in this research?

YES ☐ **NO** ☒

If YES, full ethics review required
For example:
Is there any potential that there could be physical harm for anyone involved in the research?
Is there any potential for psychological harm, discomfort or stress for anyone involved in the research?

Conflicts of interest

Do any conflicts of interest arise?

YES ☐ **NO** ☒

If YES, full ethics review required
For example:
Might research objectivity be compromised by sponsorship?
Might any issues of intellectual property or roles in research be raised?

Funding

Is your research funded externally?

YES ☐ **NO** ☒

If YES, does the funder appear on the 'currently automatically approved' list on the UTREC website?

YES ☐ **NO** ☐

If NO, you will need to submit a Funding Approval Application as per instructions on the UTREC website.

Research with animals

Does your research involve the use of living animals?

YES ☐ **NO** ☒

If YES, your proposal must be referred to the University's Animal Welfare and Ethics Committee (AWEC)

University Teaching and Research Ethics Committee (UTREC) pages
http://www.st-andrews.ac.uk/utrec/