seeCell: Visualization and Tracking Dedicated to Cell Analysis

Robin Mange[1], Pablo de Heras Ciechomski[1] and Melody Swartz[2]

[1]Visualbiotech
PSE-C EPFL, 1015 Ecublens, Switzerland
www.visualbiotech.ch
[2]Laboratory for Mechanobiology and Morphogenesis (LMBM)
LMBM EPFL, 1015 Ecublens, Switzerland

robin@visualbiotech.ch, pablo@visualbiotech.ch, melody.swartz@epfl.ch

Keywords: realtime system, computer graphics, cell tracking.

Abstract: This paper presents a method for the real-time tracking and rendering of dendritic cells in a stream of microscopes images, as implemented in the software seeCell (TM) developed by Visualbiotech (Mange, 2008). The main guideline is how computer graphics technology can increase the relevancy of the acquired data by creating new perspectives on the stream of information. Such an approach was widely used in the platform described in this paper, mostly to improve its accessibility to the scientific community.

1 Introduction

The domain of in-vivo analysis is continuously evolving and therefore the platforms dedicated to it have to progress and innovate as well. Computer graphics technology is widely used to perform this evolution since it can offer new aspects to analytic tools. Such an example is the exposition of collected results in an appropriate way to improve their relevancy and increase their meaning for the user. Moreover, using this combined power of analysis and visual worlds, it is possible to increase the interactive possibilities (see Section 2.3). These different advantages added to scientific applications permit to enhance their qualities and therefore their interest greatly. Indeed, providing a better graphic design as well as an higher level of interaction make for an easier acceptance through the scientific community. seeCell is such a platform, where the visualization and analysis domains are merged to improve accessibility and instant assimilation of complex behaviors to researchers.

seeCell was first designed for the Laboratory for Mechanobiology and Morphogenesis (LMBM) from the life science faculty of the Swiss Federal Institute of Technology (EPFL). Its goal was to help researchers to track dendritic cells in an easy and intuitive way, offering an alternative to existing tools (imageJ,). The dendritic cells are part of the mammalian immune system as explained in (Banchereau and Steinman, 1998). The development of this software was planned based on the two aspects cited above, which are the analysis and the visualization. The first obstacle encountered was how to melt them in a good product, exposing analytic contents to the user in a good and instant way. This kind of merging had to be done under a scientist's control to be sure to interpret the gathered data in the right way and to avoid any alteration of them.

2 seeCell

seeCell is composed of three main parts which are: a rendering engine, a tracking engine and an interface engine. The rendering engine's role is to visually expose the cell data extracted from the image inputs. It has also to take care of the user interface display which won't be discussed in this paper. Regarding the tracking engine, its aim is to gather the maximum of information from the cells contained in a stream of images. Eventually, the interface takes care of every kind of interaction between the user and the software such as mouse or keyboard inputs. Since the tracking task can be quite expensive, the software was designed using a multi-threading system (Steve Kleiman and Smaalders, 1996). In the current version, a main thread is taking care of the rendering and the interactions handling, while a secondary thread hosts the tracking engine. This approach allows to keep the rendering and the user inputs handling as fast as possible and independent from the image processing rate.

2.1 Visualization

The visualization engine created for seeCell has as a primary goal to increase the impact of results acquired by the tracking engine. In that way, it was opted to use an overlay of the tracked contents over the original image, following the assertion: ”what you see is what you got”. This is a good way to verify the precision of the analytic core (see Section 2.2). Moreover, this allows also to check and modify the different parameters used for the tracking process in an interactive and visual manner since you observe the changes in real-time (see sub-Section 2.2.1). Figure 1 shows an image with some dendritic cells tracked and overlapped to the initial image.

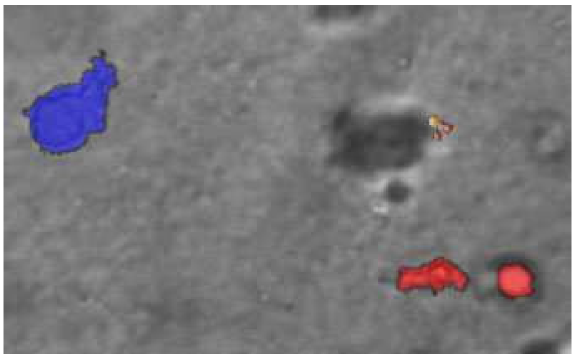


Figure 1: Three dendritic cells with their detected membrane overlayed.

This rendering pipeline is structured using layers, represented by different RGBA textures (32-bits per pixel). The alpha channel is used to create the modulation between each layer which is based on transparency. This method offers many advantages in terms of usage and simplification. First, it allows a fast computation of the cell layer since one can keep the detected entities in a per pixel basis and therefore simplify the handling of data from the tracking engine. Another advantage is the fact that working with textures is done in a per-bit basis, permitting the usage of bit level optimization (Fog, 2008). Since the final image is computed using a modulation scheme between textures directly on the main processor, no acceleration capabilities from the graphic card is required. Only the end result of the visualization is sent to the video card for screen updates. Figure 2 shows one decomposition of a final display in several layers which are the original input image, the tracked contents and another one called gradient layer which is an extra stream of data handled by seeCell and is explained later in this section.

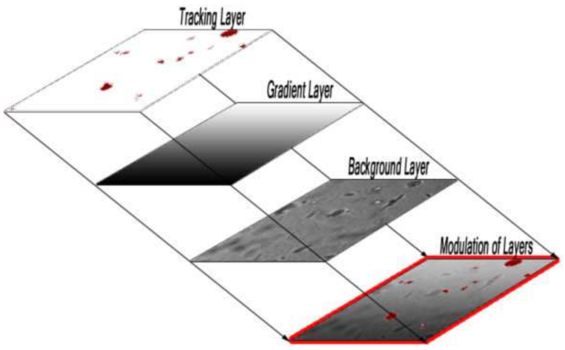


Figure 2: The rendering pipeline using Layer Modulation.

The cell layer is constructed from the tracking engine extracted data, by enabling each pixel detected as a cell using the alpha byte and setting them to a certain color depending to its state. In the general case, there are two different states for an enabled pixel: it can either belong to the outline of the cell or to its core. This decision is made by counting the number of enabled neighbors of a pixel.

Using this superposition technique, one can add extra visual features by inserting new layers. Of course like most tracking tools you can display the independent trajectories of the particles, in our case cells, but you can also display the trail left by each one of them allowing a complete pathway study by checking what was sensed and touched by whom. Figure 3 shows the trail of one cell superposed to the original image with transparency.

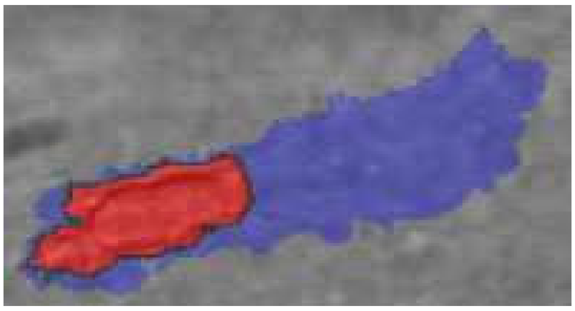


Figure 3: The trail of one cell displayed.

Coloring the reconstructed cells themselves can also bring additional information such as their membrane activity. Several special visualization mode were created focusing on certain specific aspect of the cells.

Another interesting aspect of computer graphics usage is the possibility to simulate a 3D rendering of the tracked contents in a way to get an extra dimension for trajectory analysis. It is then possible to look at the motion in x or y relative to the time which gives supplementary information. Figure 4 shows this kind of representation.

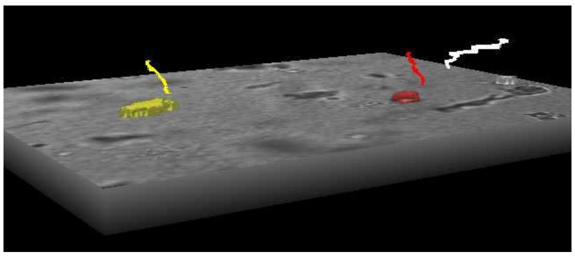


Figure 4: 3D representation of cells with their trajectories.

The Laboratory for Mechanobiology and Morphogenesis was initially interested in the study of interferences between a cell colony and a gradient generated by a chemical reaction. Indeed, the dendritic cells are sensing the world around them and move according to some criteria. By using the layer structure, it is possible to superpose another stream of images on the top of the initial ones. With this method, the user can import into the software one layer with the migration of the cells and one layer with the corresponding gradient distribution in the image. One more time, the visualization is very helpful in that case, since it helps to study the reaction of cells to certain data flow such as chemical intensity distribution over time. To clarify even more the assimilation of such behavior, the cells can be colored depending on this additional layer, showing the local gradient distribution as shown in Figure 5. You can see a cell tracked with an extra stream of data containing gradient distribution.

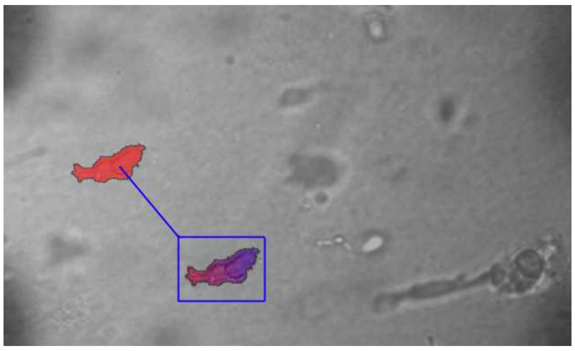


Figure 5: The cell in the box is colored from local min to max intensities according to the gradient layer loaded.

2.2 Tracking

The tracking engine is the analytic core of seeCell. Indeed, this module is used to detect cell membranes. Its accuracy is primordial since numerical stability is important. As many tracking tools (Li and Kanade, 2007), it is working in two phases which are the cell detection and the cell labeling. Since this tracking process is a complex task and can be slow, many optimization were used, such as bounding boxes (Koziara and Bicanic, 2005) to limit the number of pixels to process.

2.2.1 Cell Detection

The role of Cell detection is to find each pixel of the input image that belongs to a cell membrane. This task is achieved using mainly two different approaches. The first method is based on the creation of mathematical models, whereas the second method is based on object detection. Since the cells we want to track are alive and can deform, the first one is not applicable, therefore the second one was chosen. Two steps are required for cell detection. First an edge detection algorithm is applied allowing a partial extraction of cell outlines. Such a detection algorithm can be found in (Gonzalez and Woods, 1992). Afterwards, a hole filling method allows to finalize the detection of those membranes. Figure 6 shows the progressive result of the detection of a cell after each of these steps.

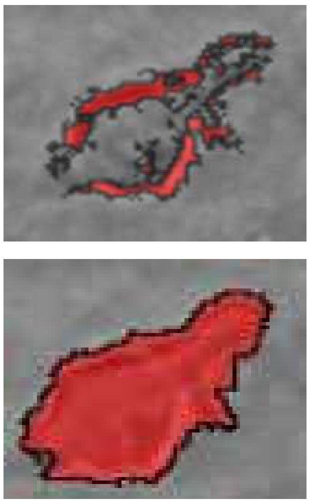
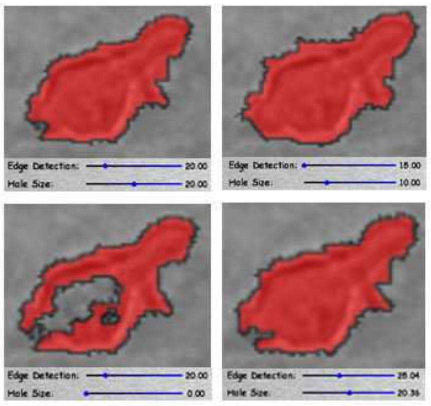


Figure 6: The results of the cell detection after each detection step.

This entire phase is responsible for the accuracy of the tool. In this end, and because a tracking engine is usually based on this approach, some parameters can be tuned by the user to optimize the process depending on the image inputs. Such values are the threshold used for the outline detection and the maximum size of holes to fill which is dependent on the size of the cells the user wants to analyze. Figure 7 shows how the membrane detection of a cell is affected when using different values for the parameters.


Edge Detection: 20.00
Hole Size: 20.00

Edge Detection: 18.00
Hole Size: 10.00

Edge Detection: 20.00
Hole Size: 0.00

Edge Detection: 25.04
Hole Size: 20.36

Figure 7: The different thresholds allow an adjustment of the tracking process.

This detection algorithm that was designed specifically for dendritic cell tracking can be changed to adapt the tool to different applications such as particles tracking or some other more specific tasks. Since seeCell was designed to be an extendable platform, it is possible to make it evolve easily to increase its possibilities, in term of tracking as well as in terms of output capabilities.

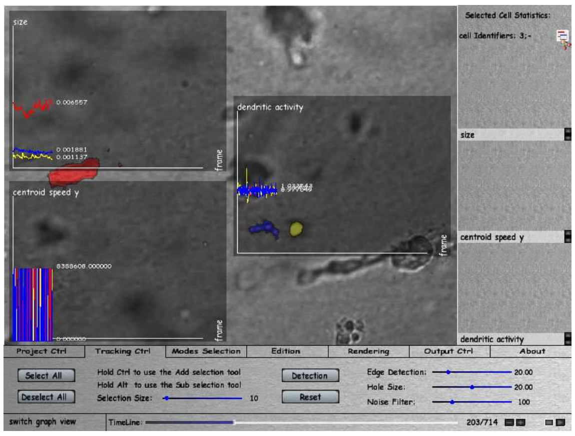
2.2.2 Cell Labeling

This second phase has for aim to assign a unique id to each cell using a propagation method. This allows to make a distinction between the different entities for the extracted information as well as for the user. Indeed, using this id, the user can select one or several cells and have it visually expressed on his screen. In the same time, he will get the corresponding statistics which are stored in memory per-cell.

This propagation method is done using a personalized stack structure to avoid any memory overflow problem than could occur with the use of a recursive method. Of course, this task is done differently if you are tracking the first frame of an image stream or an intermediate one. If you track one cell that was not detected before, you can start from any of its pixels, attribute a new label to it and propagate it to all the others. This assures to have the same id for each pixel of one cell. If the cell you want to track was already previously detected (for instance in a previous frame) you need to start the propagation algorithm from a pixel that has an old id, certifying that you will keep the same id for a cell over time. This is absolutely necessary for keeping the statistics gathered linked to the right cell.

2.3 Interactions

One important aspect of computer graphics techniques in seeCell is about interaction. Indeed, using color modulation of the independent cells, it is possible to select and deselect them to get local results. This also simplifies the understanding process of specific behaviors since they are underlined with the help of visualization. Figure 8 shows an image tracked with three plots showing the corresponding values belonging to the cells selected by the user.



Hold Ctrl to use the Add selection tool
Hold Alt to use the Sub selection tool
Selection Size: 10
Edge Detection: 20.00
Hole Size: 20.00
Noise Filter: 100
203/714

Figure 8: The plots give information about selected cells in real-time.

The different statistics that are currently gathered from each cell with the help of the tracking engine are their size, their position and speed in x and y, their dendritic or membrane activities among others. Their accuracy mainly depends on the parameters depicted in the previous section.

3 Results

seeCell in its actual state is capable to perform real-time tracking of stream of images at a rate of 30 frames per seconds in average on a dual-core 1.66GHz laptop with a Nvidia GeForce 7400 with 2GB of Ram. With the help of the multi-threading support, the rendering of the software as well as the inputs handling are always done at a rate of 60 frames per second. The accuracy of the different statistics extracted from the cells depends on the parameters used for the detection algorithms. When those are well set, an error of 1% has to be considered, which is quite satisfying.

4 Conclusion

The use of visualization techniques inside a tool such as seeCell can really improve its accessibility to the scientific community and increase the assimilation of complex processes. Regarding the tracking, there are two main problems that often occur in such applications. The first one is when an object is moving too fast relative to the frame rate since no pixels in common are found and therefore it is hard to link them. Another problem is when several objects interfere together. Indeed, when two objects merge, it is hard to distribute the pixels among them. Moreover, when they split again, it is hard to decide which cell inherits from which label. A method was found to avoid this problem, but this is not part of this paper.

Acknowledgements

Thanks to Ulrike Haessler for her support and her advices during the whole creation process of seeCell. Many thanks as well to Ivana Arsic de Heras Ciechomska for her help on tracking and signal processing algorithms.

REFERENCES

Banchereau, J. and Steinman, R. M. (1998). Dendritic cells and the control of immunity.

Fog, A. (2008). Optimizing software in c++.

Gonzalez, R. C. and Woods, R. E. (1992). Digital image processing.

imageJ. A tool including image processing and particle tracking through the particletracker plugin written by guy levy at the computational biophysics lab, eth zurich.

Koziara, T. and Bicanic, N. (2005). Bounding box collision detection.

Li, K. and Kanade, T. (2007). Cell population tracking and lineage construction using multiple-model dynamics filters and spatiotemporal optimization.

Mange, R. (2008). Rendering techniques for dynamic real-time analysis of biology. Master Thesis, EPFL.

Steve Kleiman, D. S. and Smaalders, B. (1996). Programming with threads.