APPROVAL SHEET

Title of Dissertation: Computational Methods for Rare Cell Detection in Streak Image Flow Cytometry

Name of Candidate: Miguel Roberto Ossandon
Doctor of Philosophy, 2018

Dissertation and Abstract Approved: Konstantinos Kalpakis

Konstantinos Kalpakis, Ph.D.
Associate Professor
Computer Science and Electrical Engineering Department

Date Approved: 10/30/18

ABSTRACT

Title of Document:	COMPUTATIONAL METHODS FOR RARE CELL DETECTION IN STREAK IMAGE FLOW CYTOMETRY
	Miguel Roberto Ossandon, PhD, 2018
Directed By:	Dr. Konstantinos Kalpakis, Associate Professor, Department Computer Science and Electrical Engineering

Detection and analysis of Circulating Tumor Cells (CTCs) have shown promising cancer clinical and research applications. A major challenge for the detection and analysis of CTCs is their rare nature. CTCs are found at very low concentrations in blood. Therefore, large volumes of sample are needed for meaningful enumeration, which especially impedes the analysis of CTCs using standard flow cytometry (due to its low throughput). This issue is addressed by the recent development of a high throughput imaging cytometer equipped with a wide field flow-cell. This wide-field flow cytometer adapts a technique known as "streak photography" where exposure times and flow velocities are set such that the cells are imaged as short "streaks". Streak cytometry technology enables analysis of cells in very low concentrations (0.1 cell /ml) in large volumes (10 ml) and rapidly (1 minute). However, dynamic imaging conditions in streak cytometry introduce more challenges to current automated cell counting methods, especially with low cost, low resolution webcams or smartphone cameras affordable for use in point of care and global health settings. The lack of automatic enumeration methods for streak imaging limits clinical utility of wide field streak

cytometry. In this dissertation we propose to combine traditional geometrical and intensity distribution (GID) features with visual words plus a novel image classification method based in relational features that characterize an object in relation with other objects in frames in a video file. This new cell identification/quantification method, Relational Streak Algorithm (RSA), consists of three parts: (1) finding streaks with a binary mask that contains potentially all the cells in a frame, (2) identifying candidate cells using GID features, and (3) filtering out spurious cells and identifying true cells with machine learning approaches for image classification by GID features, visual words, and relational features. We incorporate the relational features in a selective permeable filter that can either discard cells, allow cells to proceed through the filtering layers, or defer the cells to the most sensitive classifier (in this case the visual words classifier) for final classification. We evaluated the RSA using samples with nominal concentrations of 1 cell per mL and 1 cell per 10 mL (consistent with acceptable numbers of CTCs considered to show clinical significance). The RSA performed well with both concentrations. In the 1 cell per mL dataset, the algorithm achieved 88% sensitivity with an F1 score of 91%. In the 1 cell per 10 mL dataset, the algorithm achieved sensitivity of 84% with an F1 score of 75%, outperforming earlier versions of the algorithm and current tools for cell tracking (CellTrack and MTrack2) used as comparisons. These findings demonstrate superiority of the new analytical capabilities of streak-based cytometry when coupled with the RSA for automated cell detection and counting. This cell counting capability enables automated low-cost streak imaging flow cytometry detector for clinical and research use and offers the possibility of expansion of cell-based clinical diagnostics to resource-poor settings.

COMPUTATIONAL METHODS FOR RARE CELL DETECTION IN STREAK IMAGE FLOW CYTOMETRY

By

Miguel R. Ossandon

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, Baltimore County, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2018

Dedication

This dissertation is dedicated to the two most important women in my life. My mother, Rosa Eliana that inspired me, and my wife and friend, Melanie, with her endless patience, love and constant encouragement, I was able to complete this journey.

Acknowledgements

Nine years ago, I started my career as a doctoral student. This period has been an intense learning experience with a big impact on me, not only academically but also my personal life. Important things happened in my life during this time, but I would like to mention the happiest and saddest. I got married, but I also had a loss. My brother died of lung cancer around six months after I enrolled for my first class. During this time my extended family, mother and father in law Bob and Pam, and my wife, Melanie, gave me the strength to press on. Therefore, first of all, I would like to thank my wife and our family for their cheerfulness, patience and continual support.

I must recognize that many other people contributed and helped me in this journey with their advice and encouragement. Without them I could not have accomplished it.

I would like to thank my dear friend, Avraham Rasooly, for his encouragement to pursue my doctoral degree in the first place and for his help and guidance in the direction of my research that ended up in a successful thesis.

I would also like to thank my friends and colleagues at the National Cancer Institute (NCI), Houston Baker, Brian S. Sorg, Rao Divi, George Redmond, Luis Alejandro Salicrup and Lokesh Agrawal for their willingness to help and for their advice.

Having a full-time job at the NCI during my intensive academic years made it critical to have the support of my supervisors. I am extremally grateful to James W. Jacobson, Sheila Taube, Barbara A. Conley and James V. Tricoli for their support. In memory of Dr. Jacobson, I would like to say that his kindness and respect that he showed for everybody inspired me. I am very grateful for his encouragement when I started this journey, unfortunately, he is unable to see how it ended.

I cannot miss this opportunity to thank the members of my dissertation committee Professor Chein-I Chang and Professor Dhananjay Phatak for their input and time invested to instruct me and review my work.

Particularly, and most importantly, I want to thank my advisor, mentor and friend, Professor Konstantinos Kalpakis. Without his guidance, I would not have succeeded in this endeavor.

Above all, I want to thank God for giving me the energy and resilience to persevere in my career as a doctoral student and for surrounding me with such a group of friends that always encouraged me to go on.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Circulating tumor cells

When a primary tumor reaches a critical mass, it sheds tumor cells that migrate into blood vessels and circulate in the bloodstream. These rare cells, called circulating tumor cells (CTCs), are responsible for development of distant cancer metastases. Detection and analysis of CTCs have shown promising applications for cancer care and clinical cancer research including cancer prediction, prognosis, early detection, guidance in treatment selection, disease monitoring and surveillance, among others. Recently, CTC technology has matured enough to achieve acceptable reproducibility and sensitivity levels to explore clinical utility [1-4]. Detection of CTCs, however, is challenging since they are found in very low concentrations in blood and need large sample volumes for meaningful detection and enumeration. This limitation impedes the analysis of CTCs by current standard flow cytometry, which has sparked the interest of the scientific community to develop technologies to address this issue [5-9].

Since CTCs can be released by a tumor in early stage cancer, a great potential application of CTCs is in cancer prevention. This is particularly important in low and middle income countries (LMICs) where cancer is usually diagnosed in an advanced stage, which limits treatment options[10]. However, technologies for cell detection are not usually suitable for low resource settings such as found in LMICs. CTC technology

suitable for low resources settings must be affordable, portable and designed to work with minimal infrastructure[11-15].

1.2 Medical diagnostic technologies for low resource settings

Over 80% of the population of the world live in low and middle income countries (LMICs) and more than 700 million people live on less than $1.9 a day[16,17]. In addition, cancer is one of the leading cause of death in the world (second after heart disease in the US) with around 70% of the cancer death occurring in LMICs where inaccessible or late stage diagnosis, poor prognosis and/or lack of treatment alternatives is common [10]. Diagnostic cancer technologies developed for high income countries are often not suitable for low resource settings, either because they are not affordable or not compatible with needs and conditions in LMICs, consequently development of affordable technology for low resources settings, including analysis of CTCs, is challenging [11-14]. In order to be suitable for use in low resource settings, CTCs as other diagnostic technologies must be designed to work with minimal medical infrastructure and limited access to health care along with portability, ease of use, local expertise and readily availability of components for troubleshooting [12,15].

Lab-on-a-chip (LOC) technology provides a potential approach for the development of Point of care multichannel detection analytical tools suitable for low resources settings [18-22]. LOC has facilitated the implementation of chemical and biological assays outside laboratory environments [23]. Optical detectors using charged-coupled device (CCD) or

complementary metal-oxide-semi-conductor (CMOS) cameras coupled to a microfluidic chip have been proposed for development of biosensor array allowing many areas of the sample to be interrogated simultaneously[24-27].

Many CCD-based LOC detection systems for multiple applications such as immunodetection and food poison applications; and infections agents have been developed with sensitivity compatible with their current laboratory counterparts. However, these LOCs devises uses expensive state-of-the-art technology, for example cooled CCD cameras such as the ones used for astronomy applications limit their use in LMICs due to prohibitive cost [19,20,24-26,28-31].

Many LOCs for microcopy applications have also been developed to address the issue of cost, including wide-field microscopy where the regular optical lenses are replaced by fiber-optics. These lensless-on-a-chip microscopes are lightweight, portable and can achieve high resolution and are suitable for use in telemedicine applications [32-40]. This technology can be coupled to a microfluidic device for detection and analysis of CTCs in LMICs, but the microfluidic is still limited by low flow rate impeding the system to analyze large volumes of samples (necessary for rare cells detection). In addition, these systems use high speed imaging to capture the motion of the cells producing large files requiring more computer power for management and the cells are imaged as a small group of pixels that are difficult to distinguish from noise.

Mobile phones have been adapted for biodetection, integrating image capabilities and communication, and can be effectively used for mobile Point-of-Care (POC) devices

in mobile health (mHealth) applications[41-51]. However, the cameras in these phones often have high noise levels and they are also less versatile with their optical systems than webcams (e.g., inability to change lenses). Cytometry in a cell phone has been described (optofluidic fluorescence cytometry) as alternative for remote and resources-limited environments. While mobile and versatile, the flow rate of this system is ~1 ul/min, which also limits the analysis to small volumes.

A high throughput microfluidic cytometer, the streak-mode wide field flow cytometer, was developed to address the issue of large sample volume limitation on standard flow cytometry for the analysis of CTCs, and to address technical difficulties that would impair utility in low resources settings [52,53]. This device uses a wide field flow-cell instead of the conventional narrow cell used in traditional cytometry, which enables the analysis of large volumes at low flow rates in brief, clinically useful periods of time. The wide field cytometer adopts a technique used in Particle Image Velocimetry (PIV) known as "streak photography" where exposure times and flow velocities are set such that the particle images are displayed as short streaks. Since the streaks are imaged over a large number of pixels, they become more easily distinguishable from noise that appears as speckles, a difference that increases the sensitivity of the device enough to enable the use of low sensitivity webcams or mobile phone cameras (mobile phones are extensively used in LMICs) and make the device suitable for low resource settings[52,53]. However, in the initial method the cells were count manually making the technology not practical for clinical applications, and no method for cell identification

and enumeration specific for streak image cytometry (a barrier to the translation of the device to clinical settings) has been published.

1.3 Aims and motivation

The main aim of this dissertation is to describe our development of a method for quantitative analysis of cells by streak image cytometry with automated cell counting that will make streak imaging cytometry useful for research and clinical applications.

This dissertation describes the development of a relational streak image detection algorithm to improve the detection level, reproducibility and to make the technology operational for research or clinical settings, especially for LMICs.

The algorithm combines geometrical and intensity distribution features with relational features and visual words for the detection and enumeration of streak images of single cells. One important contribution of this technology is its unique capacity for analyzing very small cell concentrations (0.1 cell /ml) in large volumes (10 ml) in short times (1 minute) which enable high throughput analysis of circulating tumor cells. This cell counting capacity enables automated low-cost streak imaging flow cytometry based on a CCD detector, and offers the possibility of expanding cell-based clinical diagnostics to resource-poor settings. The wide-field streak image cytometer and our relational streak detection algorithm developed in this thesis provides a simple, affordable, and portable cytometer which we expect will facilitate the expansion of cell-based

diagnosis into low resources settings. This technology could also be implemented in other rare cell applications beyond circulating tumor cells, such as analysis of immune cells, e.g., T-cells, B-cells, hematopoietic stem cells, or rare cells related to respiratory disease where bulk cell population analysis fails to characterize rare cell signatures by diluting the contributions of rare target cells. The high throughput characteristic of this technology also could be used in applications beyond rare cell analysis, such as bacterial contaminants in water or food, or rapid analysis of bacterial susceptibility to antibiotics, etc. These potential applications will be discussed in more detail in chapter 10 (Future Development and Potential Applications).

The rest of this dissertation is organized as follows. Chapter 2 provides background information and a literature review of CTC biology, technology and current CTC detection methods. Chapters 3 and 4 focuses on common computational algorithms useful for cell image detection and tracking. Chapter 5 includes information about sample preparation and ground truth. Chapter 6 introduces the first algorithm developed for streak image cytometry. Chapters 7 and 8 describe the development of the relational streak algorithm and its evaluation. Chapter 9 discusses results and provides points for discussion and future development. Finally, chapter 10, as mentioned above, discusses limitations of the technology, future developments and potential applications beyond CTCs.

Chapter 2: Circulating Tumor Cells

2.1 Introduction

Tumor development alters local metabolism producing physiological changes in the tumor microenvironment. For example, high metabolism, which is the characteristic of many cancers, produces early changes in levels of oxygenation around the tumor followed by a decrease in oxygenation in the central part of the tumor often causing central necrosis[54]. The tumor cell around the tumor becomes more metabolically active and when the tumor reaches a critical mass, tumor cells shed from the primary tumor and migrate into the circulatory system and travel through the bloodstream or lymph vessels [3]. These rare cells called circulating tumor cells (CTCs) cross through the wall of vessels resulting in systemic dissemination and extravasation in distant parts of the body producing of distant cancer metastases (Figure 2.1).

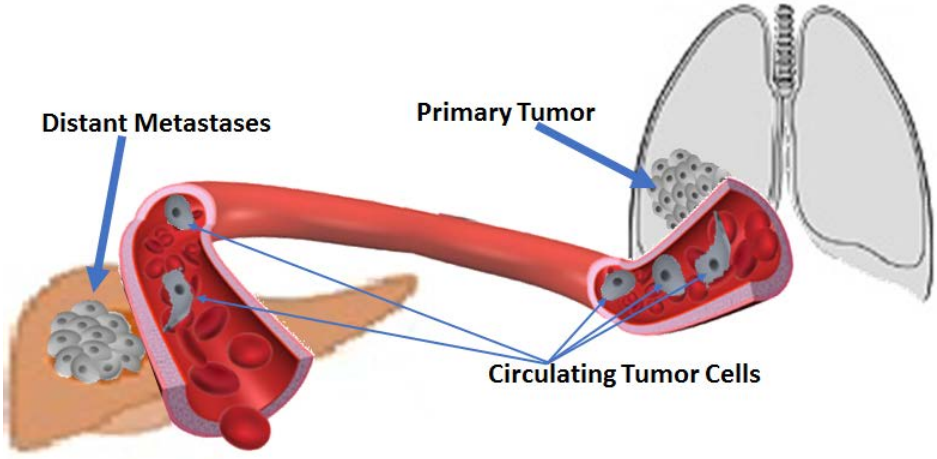


Figure 2.1: Circulating Tumor Cells*. CTCs are shed from the primary tumor, travel through the circulatory system and cleave in distant organs producing cancer metastases.*

CTCs were first reported in 1869 in the blood of patients with subcutaneous tumors, but only after recent advances in cell technology, their identification and characterization has been made possible [55]. Subpopulations of CTCs with tumor-initiating potential can act as mediator for micro-metastases that can eventually generate a larger lesion. Nowadays, the clinical applications of CTCs include early detection, diagnosis, prognosis, treatment monitoring, identification of therapeutic targets and evaluating treatment efficacy (to help patients from being exposed to ineffective therapeutics) [1-4,56,57].

Death of cancer patients is rarely caused by the primary tumor and cancer metastases account for most of cancer related deaths (nearly 90% of cancer deaths occur due to cancer metastases). This could be attributed to the fact that the treatment decisions are usually based on the characteristics of the primary tumor (since biopsy of metastatic lesion are difficult to obtain), but the metastatic lesion shows high heterogeneity compared with the original tumor resulting to cancer patients may receive a suboptimal treatment. Therefore, there is a need for technologies to better evaluate metastatic cancers [1,3,55,57-60] in this area. Characterization of CTCs through genetic signatures, or physiological or physical features to identify CTCs that are capable of metastases (among from the ones that are not) and understanding the metastatic process is an active area of cancer research.

2.2 Circulating tumor cells sample size and sampling probability

The detection of CTCs in peripheral blood is challenging since CTCs occur at very low concentration. CTCs are found in frequency of 1-10 cells per ml of whole blood in metastatic patients, (~1 CTC per 10^6 – 10^8 white blood cells (WBC)), this frequency reduce in half in non-metastatic patients [8,61]. However, the significant number of CTCs per ml of blood associated with poor prognosis is not well defined, in addition, the frequency of CTCs obtained from a sample of blood may not reflect the cell population [60]. For example, CellSearch data traditional suggested that 5 CTCs per 7.5 ml of blood is associated with poor prognosis, but after of reanalysis of published CellSearch data was found that the CTC threshold for identification of patients with poor prognosis can be potentially reduced from the current value of 5 cells to 1cell per 7.5 ml as clinical significant [62]. In addition, since the proportion of CTCs in different cancers is not uniform, the determination of a cut off number of CTCs per ml of blood to stablish prognosis and disease outcome is challenging. For example, a re-analysis of CellSearch data from healthy donor, patients with benign conditions and patients with metastatic cancer (Breast, Colorectal and Prostate) shows the variability of the percent of patients with CTCs over cut off values of 1, 5, 10, 50, 100, 500 and 1000 CTCs per 7.5 ml of blood (Table 2.1) [63].

Table 2.1: Frequency of CTC in 7.5 mL of blood from normal donors, patients with benign disease, and patients with metastatic breast, colorectal, and prostate cancer before initiation of a new line therapy. The numbers represent the percentage of patients with CTC above threshold [63].

	# Patients	=1 CTC	=5 CTCs	=10 CTCs	=50 CTCs	=100 CTCs	=500 CTCs	=1000 CTCs
Normal	295	3.4%	0%	0%	0%	0%	0%	0%
Benign	255	7.5%	0.4%	0.4%	0%	0%	0%	0%
Breast	177	70.6%	49.7%	38.4%	20.9%	15.8%	3.4%	2.8%
Colorectal	413	47.5%	18.2%	11.6%	2.4%	1%	0%	0%
Prostate	218	77.5%	57.3%	45%	20.6%	13.8%	3.7%	2.3%

To put the this results in the perspective of this thesis, the concentration of CTCs in blood was estimated using the data from Table 2.1 considering samples with at least 1, 5, 10, 50, 100, 500 and 1000 CTCs per 7.5 ml of blood, as well as the number of patients (N) for each of the normal, benign and metastatic breast, prostate, and colorectal cancer. For this estimation, cell counts are assumed to follow Poisson distribution. The CTC concentration ρ was estimated using the Maximum Likelihood method:

$$\operatorname{argmax}_{\rho}\left\{\prod_{i=1}^{8}\left(\sum_{j=l_i}^{u_i}(\rho V)^{j}e^{-\rho V}/j!\right)^{n_i}\right\}$$

Where (l_i) = [0, 1, 5, 10, 50, 100, 500, 1000], (u_i) = [0, 4, 9, 49, 99, 499, 999, ∞], and n_i is the number of subjects with $[l_i, u_i]$ CTCs in V=7.5 ml.

The estimation was done using MATLAB 2016b with the Newton-Raphson algorithm to maximize numerically the symbolic log-likelihood, whose numerical evaluation required 1033 decimal digits of precision to handle numerical underflows. The results are given in Table 2.2 Table 2.2 shows that healthy donors and individuals with benign

condition have less than 0.005 CTC/ml. Breast and prostate metastatic patients have about 1 CTC/ml, while metastatic colorectal cancer patients have about 0.07 CTC/ml.

Table 2.2: Maximum Likelihood Estimation of the number of CTCs in blood. Estimation is done assuming a Poisson cell count process and using the CTCs experimentally detected by the CellSearch system in 7.5ml blood samples taken from patients that are healthy/benign or have metastatic cancers.

Type	CTCs in 7.5 ml of blood	CTCs in 1 ml of blood
Healthy Individual	0.00461	0.000615
Benign Condition	0.01536	0.002048
Metastatic Breast Cancer	7.51387	1.001849
Metastatic Colorectal Cancer	0.51784	0.069045
Metastatic Prostate Cancer	7.64334	1.019112

Another challenge for detection of CTCs is heterogeneity and the lack of cell markers expressed for all CTCs. For example, a common biomarker used to identify CTCs is Epithelial Cell Adhesion Molecule (EpCAM), however, some CTCs lose EpCAM in the transition from epithelia cells to a more invasive/aggressive form, through the epithelial–mesenchymal transition process, and may not express EpCAM [8,64].

2.3 Current commercial methods for isolation of CTCs

Many research and commercial platforms have been developed for enumeration, capturing and isolation of CTCs, a brief review of some of these methods representing different platforms are provided in this section.

2.3.1 Methods based on immunoaffinity

Many methods have been developed based on antibodies targeting tumor-associated surface antigens to capture CTCs such as the CTC-Chip and the Herringbone-Chip, CellSearch, GEM (based on microvortices mix), Ephesia, MagSweeper, and CellCollector (in vivo isolation of CTCs). A sample of these technologies are briefly described below.

CellSearch® System: CellSearch® (Veridex LLC, Raritan, NJ, USA, developed in the early 2000s) is the only FDA approved technology for enumerating CTCs in the clinic as aids for prognosis and treatment monitoring for metastatic breast cancer, prostate cancer and metastatic colorectal cancer[8,65]. The CTCs are enriched by positive selection using EpCAM expressed on the surface of the cells as a target. The blood sample is mixed with magnetic iron nanoparticles attached to anti-EpCAM antibodies and the CTCs are isolated through an electromagnetic field. The enrichment step is followed by immunofluorescent staining for CK and positive CTCS identification by semi-automated fluorescence microscopy. Five or more CTCs in 7.5 ml of blood indicated a bad prognosis for breast cancer or prostate cancer patients (three or more for colorectal cancer patients) [7,55,65].

MagSweeper™: This device was developed at Stanford University. CellSearch® enriches CTCs using epithelial cell surface markers such as EpCAM and immunomagnetic beads. The sample containing the CTCs are incubated with

antibodies attached to magnetic beads (similar to CellSearch®) and loaded into the sample wells. Round-bottom, neodymium, magnetic rods (robotically driven) sweep through the wells in overlapping concentric circular loops covering the entire well area while applying a magnetic force to capture the CTCs. The capture cells attached to the rods are washed with a buffer and the rods (with the attached cells) are robotically moved to the release wells where the magnetic field is removed, and the cells are released. Several rounds of capture-wash-release-recapture can be applied to eliminate normal cells or any other cells not specifically attached to magnetic particles. This device can process 9 ml of blood per hour and is able to capture more than 50% of the labeled CTCs [8,66].

Cell Collector® (GILUPI nanomedizin, Netzwerk Diagnostik Berlin, Germany): The Cell Collector is a *in vivo* technology for isolation of CTCs. The enrichment is achieved using a stainless-steel medical wire (Seldinger guide wire), with a 20 mm tip covered with a 2 μm thick gold layer. The wire is functionalized with anti-EpCAM antibodies attached to the gold layer through synthetic polycarboxylate. Using a conventional 20G intravenous cannula, the guidewire is inserted into the cubital vein exposing the functional gold tip to the blood allowing CTCs to be captured. The guide wire is left inside the vein for 30 minutes (screening around 1.5 liters of blood) and then retrieved for CTCs analysis through immunofluorescence or Polymerase Chain Reaction (PCR). Using this technique, CTCs were successfully enriched from the blood of breast cancer and non-small-cell lung cancer (NSCLC) patients[8,67].

2.3.2 Methods based on biophysical properties

These methods use physical properties such as density (AccuCyte), size (Micro Spring Array, ClearCell), size/deformability (ISET, previously described), or electrical properties (DEPArray) or an electrical signature (ApoStream) for isolation of CTCs.

AccuCyte® – CyteFinder® System (RareCyte, Inc, Seattle WA, US): Uses a density-based enrichment method (AccuCyte) combined with digital scanning microscopy (CyteFinder). Different fractions of the blood are separated by centrifugation using the AccuCyte kit. The white cell layer (where the CTCs are found) is extracted and spread onto 8 SuperFrost®-Plus microscope slides (150 µL per slide). Then, the cells on the slides are fluorescence stained and analyzed under a 4-channel digital scanning microscope (CyteFinder). Using a computer software, the objects are classified as (1) "Cell," (2) "Not a Cell" and (3) "Indeterminate," according to the fluorescence signature produced by the surface antigens. The CTCs are identified for surface markers such as EpCAM. The identified CTCs can be extracted using a single-cell retrieval device (CytePicker™) for subsequent analysis by PCR[8,68].

ClearCell® FX1 System (Clearbridge BioMedics): ClearCell takes advantage of hydrodynamic forces (inertial lift and Dean drag forces), cell size and inertial movement of cells in curvilinear microchannels for cell separation. In this device, cells are under the influence of inertial lift force and Dean drag force. Neutral buoyant particles under the influence of inertial lift force arise from mainstream and move away from the center of the channel towards the channel's walls. In addition, the curvilinear

nature of the channels introduces a secondary rotational flow-field perpendicular to the flow direction producing a drag force. The balance of these forces would determine the location of the cells as a function of their size; cells of distinct size will migrate to distinct lateral positions near the channel wall allowing size-based separation [69-71].

DEPArray® System: DEPArray® (Menarini Silicon Biosystems, Bologna, Italy) is based on a microfluidic chip configured by an array of individual controllable electrodes with embedded sensors that allow the creation of a dielectrophoretic (DEP) force enabling the capture of cells in DEP cages. The captured cells are scanned and imaged by an automated fluorescence microscope, the cell images are analyzed for morphological features and staining patters to discriminate between tumor and normal cell. The identified CTCs can be moved to the recovery chamber and deployed to a tube outside the chip through changing the DEP field around the cell. This system was used for successful detection and recovery of CTCs in colon cancer [7,8,72,73].

2.3.3 Microfluidic based analysis of CTCs

Microfluidic devices have been extensively used for enumeration and isolations of CTCs. Since microfluidics can be designed to produce minimal shear stress and cells do not require fixative for processing, CTCs can be isolated with minimal damage[74]. Many microfluidic techniques are based on surface cell makers identified by antibodies, but the efficiency of this approach is limited to the sensitivity of the antibody[75]. In traditional microfluidic devices, cells follow streamlines due to

laminar/uniaxial flow conditions. This lack of mixing results in limited interactions between cells and antibodies amplifying the problem of antibody sensitivity [74]. To address this issue of limited cell-antibody interaction, novel microfluidic designs have been reported. For example, the integration of a microposts array functionalized with antibodies (e.g. anti-EpCAM) into the chip can be used to break up the streamline flow and enhance cell-antibody interactions; this approach was implemented in the CTC-chip [76]. CTC-chip was an improvement from traditional microfluidic chips, but still relies on laminar flow and is dependent on a complex microposts structure difficult to scale up for production[74]. An alternative to address this issue is to design the walls of the microchannels with surface ridges or herringbones to produce turbulence and disrupt the streamline flow increasing the interaction between cells and the antibody-coated walls(Figure 2.2). This approach combined with immunomagnetic particles or other methods has been shown to effectively identify and isolate CTCs [74,77-79].

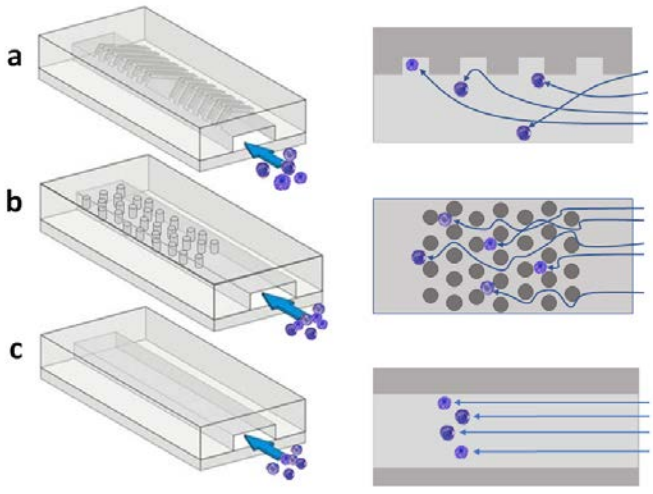


Figure 2.2: Representation of three distinctive designs of microfluidic channels*: c) Tradition design channel resulting in laminal flow conditions. b) Microposts array structure that enables to break up the streamline flow and enhance cell-antibody interactions (CTC-chip). a) Herringbone walls produce turbulence and disrupt the streamline flow (adapted from [74]).*

Another alternative to improve capture efficiency of CTCs is replace antibodies by aptamers. Aptamers (single-stranded nucleic acid oligomers) have high affinity for molecular targets with faster tissue penetration compared with antibodies and have been showed to successfully capture and enumerate CTCs [75]. However, as antibodies, they are limited to the sensitivity and availability of the aptamer.

Reverse transcriptase polymerase chain reaction (RT-PCR) and quantitative real-time RT-PCR have been also used to indirectly identify CTCs through the detection of the expression of target genes in peripheral blood and other bodily fluid in patients with different cancer types such as metastatic breast, colorectal, gastric and bladder cancer among others [80-83]. RT-PCR has shown high sensitivity in detection of CTCs, but one issue is the destructive nature of the sample processing. For RNA extraction (preliminary step for RT-PCR), cells must be homogenized impeding RT-PCR to be use for enumeration or further morphological analysis of CTCs [84].

One alternative to the use of cell markers or nucleic acid for CTCs identification and isolation is to use physical characteristic of the CTCs such as size or deformability. CTCs are usually larger than normal cells and many microfabricated filtering systems have been developed to take advantage of this characteristic. Size detection methods are label-free, simple and fast and allow morphological and genetic characterization of individual cells [85]. One issue for size separation is that CTCs are highly deformable and must go through a chemical fixation process to be stiffened otherwise they can squeeze themselves through filter pores, but stiffed cells are difficult to elute from filter pores

making it difficult for downstream analysis. In addition, upon fixation, cells are no longer viable (cannot be cultured) [86].

Several methods have been proposed to avoid the fixation step before filtration. For example, ISET (Isolation by Size of Epithelial Tumor Cells, developed through funding from the French National Institute of Health and Medical Research), was reported to successfully isolate CTCs in blood from patients with cutaneous Melanoma. ISET uses a filtration system based on a polycarbonate membrane with calibrated 8-μm-diameter cylindrical pores. ISET can achieve isolation of CTCs (without cell fixation) applying gently negative pressure producing a vacuum over the filtration system[84,87,88]. In addition of the need for cell fixation, size separation, faces other challenges with small CTCs since they can still fall through the filter. Large white normal cells can also be a problem, since they can be enriched as the CTCs and contaminate the CTC population yielding limited purity. In addition, direct filtration systems are prone to clogging, amplifying the issue of low throughput associated with CTCs separation by size [9,89].

2.4 Flow cytometry and CTCs analysis

While the most common technique for cell detection and counting is flow cytometry, it has not been used for CTC analysis. Flow cytometry utilizes microfluidic sheathing and integration of optic through a technique based in hydrodynamic focusing[90]. Briefly, a central channel is enclosed by an outer sheath of fluid with faster flow. As the outer fluid moves, it creates a drag force in the central channel increasing the velocity in the

center and reducing the velocity in the walls. The center flow stream is compressed between two sheath streams producing a single line of particles/cells (Figure 2.3). In flow cytometry, cells are labeled with fluorophores and exited by a laser allowing that individual cell to be interrogated one at a time. Flow cytometry can interrogate a large number of cells in a short time providing quantitative information based on the cell's fluorescent signature and multiparameter analysis in cell populations [91].

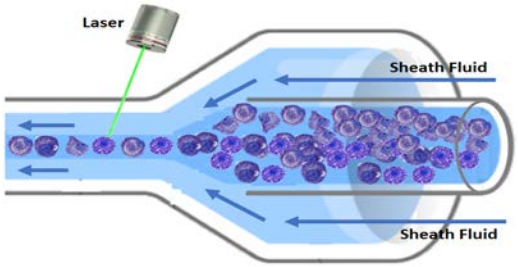


Figure 2.3: Hydrodynamic focusing*. Using microfluidic sheathing the cells are aligned in a single line, and individual cells can be interrogated by the laser (figure adapted from https://www.thermofisher.com).*

The latest technologies in microfluidic sheeting include passive hydrodynamic focusing generated by chevron grooves embedded on the walls of the microchannel. This allows the sheath fluid to surround the stream enabling the analysis of thousands of particles per second [90,92].

One problem with hydrodynamic focusing is a low flow rate due to high hydrodynamic resistance of the cell. For example, standard cytometers have a flow rate around 10-20 µL/min[93], limiting the device to small volume samples or long analysis, impeding this technology for analysis of rare cells which require large sample volumes since CTCs

are in extremely low concentrations[94]. Higher flow rates (100-1000 µL/min) have been reported in cytometers using sound waves to focus the cells into a single cell flow in the cytometer, a process called "Acoustic-assisted hydrodynamic focusing." This method minimizes the impact of higher pressure over the fluidic and allows the cells to remain confined in a narrow region eliminating issues associated with increased flow rate in traditional cytometry [93].

Other approaches have been proposed to address the challenges of large volume analysis in flow cytometry including combining full field imaging sensors (as optical detector) with classical flow cytometry [91]. One way to address this issue of low number of cells is to perform enrichment of CTCs before to flow cytometry detection using positive and/or negative selection (many techniques use a combination of both). Negative selection aims to remove normal cell from the CTC population including red and white blood cells. Red blood cells can be eliminated using a lysis buffer. White cells can be isolated using biomarkers expressed by normal white cells, but not expressed by CTCs such as CD45 or CD66 or methods that combine density gradient centrifugation with biomarker-based enrichment [8,95]. By contrast, positive selection (more widely used than negative selections) aims to capture and isolate CTCs from the total cell population[95]. This can be achieved using physical properties of the CTCs, (e.g. size, density electrical charge or deformability) or biological properties using immune markers expressed on the surface of the CTCs such as EpCAM, N-cadherin, Cytokeratin, EGFR and others [8,55,61]

2.5 Limitations of current technologies for analysis of CTCs

There are two main limitations of current methodology for the analysis of CTCs [60]:
(i) Complexity of the device that could make it not suitable for Point-of-care (POC).
(ii) Sample volume needed for accurate analysis.

Point of Care for CTCs analysis: POC technology has the potential to play a significant role in patient diagnosis and management especially in LMICs. POC assays must be simple, convenient, affordable, and provide rapid results. For example, CellSearch (the only FDA approved CTC method) employs a complex multi-step sample preparation method and extensive human interaction for identification of intact/damaged CTCs. This requires a trained expert for the interpretation of individual CTCs (based on established selection guidelines) and potentially introduces operator variability into test results [60].

Sample volume needed for analysis: LoC technology can be suitable for POC settings, but as previously discussed the rare nature of the CTCs introduce a difficult challenge for cell detection. Inconsistent results have been reported among different CTC assays, including LoC methods, pertaining detection rate [60]. Detection rate refers to the number of CTCs detected divided by the total number of CTCs in the whole sample. Since only few CTCs can be found in a blood sample, missing as little as one cell can considerably decrease the detection rate.

The detection rate can be increased by increasing the sample volume within the clinically allowable range, for example detectable CTCs were found for 20% more patients when using 30 ml instead of 7.5 ml of blood. The volume of blood was also found to be a limiting factor in flow cytometry-based CTC assays[60]. However, increasing the sample volume to 30 ml can increase the detection time to levels incompatible with clinical practice in POC settings where high-throughput technology is required. As discussed above, LoC devices are compatible with POC settings, but current LoC technology for CTCs detection suffers from throughput too low for meaningful CTCs enumeration. Table 2.3 shows the flow rates of current technologies for CTC detection, which highlights the slow rate of modern microfluidic systems [96].

Table 2.3: Rare cell detection technologies and flow rate[6].

Technology	***Detection Method***	***Flow rate***
CellSearch (FDA approved)	EpCAM/magnetic beads	-
MagSweeper	EpCAM/magnetic beads	9 ml per hour
CTC-Chip	EpCAM/magnetic post	1-2 ml per hour
Herringbone Chip	EpCAM/microvortices mix/magnetic beads	4.8 ml per hour
GEM	EpCAM/microvortices mix/magnetic beads	3.6 ml per hour
Ephesia	EpCAM/magnetic beads	3 ml per hour
Accucyte	Density	-
ISET	Size/deformability	-
DEPArray	dielectrophoresis	-
ApoStream	Electric signature	10 ml per hour
ClearCell	Size/hydrodynamic forces	1-1.5 ml per minute
Streak Cytometry	Immunoidentification/streak imaging	10 ml per minute

2.6 Streak cytometry

To address the limitations described above (Sample volume and POC) for rare cell detection specially in LMICs, a wide field streak flow cytometer for high volume throughput analysis (Table 2.3) has been developed [52,53]. The wide field streak flow cytometer is based on video imaging flow cytometer integrated with a wide flow cell for high throughput that allows it to interrogate a large sample volume in brief time. The device is composed by a 1W 450 nm laser used for fluorescent excitation (Hangzhou Brand New Technology Co., Zhejiang, China). The fluorescent emission is detected using a green emission filter with center wavelength 535 nm and bandwidth 50 nm (Chroma Technology Corp., Rockingham, VT). A Sony PlayStation Eye webcam equipped with a c-mount CCTV lens (Pentax 12 mm f/1.2) is used as the photodetector. The webcam sensor is connected to a 32-bit Windows-based laptop computer via a USB2 port (Figure 2.4). The camera control software is used to set camera parameters (exposure time, frame rate and gain) and to save video in an uncompressed AVI format. The target cells are fluorescently tagged, and the fluorophores are excited by the laser, and imaged by the web-cam (Figure 2.4). The signal can also be analyzed by a cell-phone or tablet, keeping the system affordable, portable and easy to use. A key component of this devise is the wide, high throughput, flow cell that is used instead of the conventional narrow hydrodynamic focusing cells used in traditional flow cytometry.

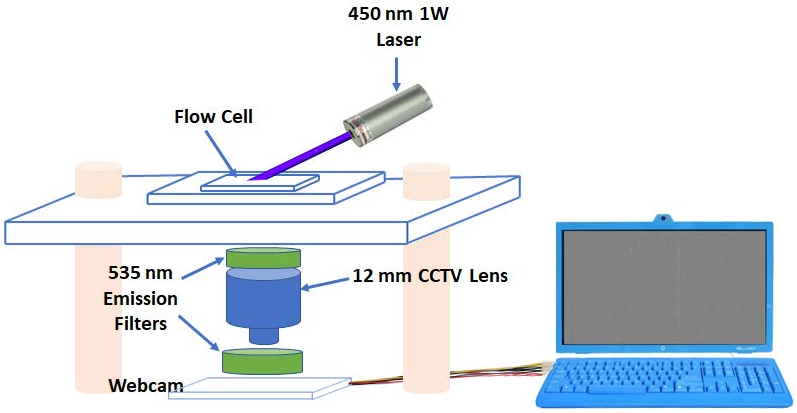


Figure 2.4: Schematic of the wide-field cytometer*. The sensing element includes a 12 mm f/1.2 CCTV lens and two green filters. The excitation source is a 450 nm 1 W laser.*

The channel of the flow cell was widened to 20 mm to maximize enabling the analysis of higher volumes (e.g., 20 ml instead of 20 ul) at lower flow rates, which is not possible with most conventional flow cytometers. To provide uniform excitation across the width of the channel, the laser source is injected into the side of the flow cell such that it forms a linear band of excitation across the center of the field of view (Figure 2.5).

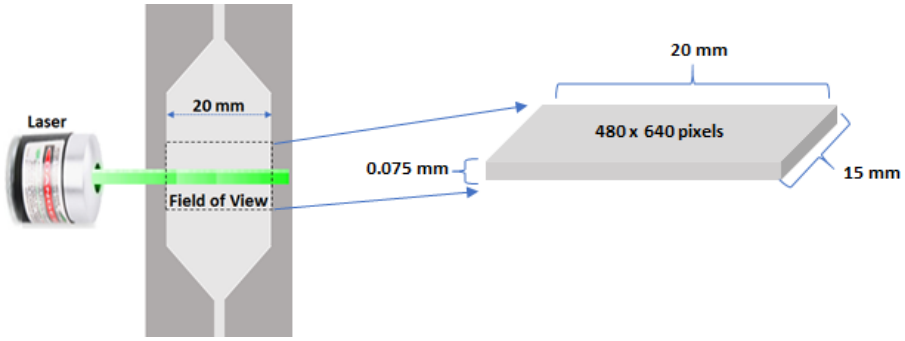


Figure 2.5: Wide view flow cell*. The wider flow cell allows analysis of higher volumens.*

The image modality for the wide-field flow cytometer is based on a technique used in Particle Image Velocimetry (PIV) known as "streak photography." Streak photography has been reported since 1950s for illustrating motion in fluid; the term "Particle Image

Velocimetry" first appeared in the literature in 1984 [97,98]. In streak photography, the particles are illuminated by a continuous light source and exposure times and flow velocities are set such that the particles are imaged as short "streaks." In this case the high flow rate moving cells are imaged in low frame rate exposure so that the path of the moving cells results in a "streak," (Figure 2.6) whose length is proportional to the exposure time [94,97]. Since streaks are imaged with a large number of pixels, they are easily distinguished from the noise, which appears as "speckles." This increases the detection capabilities of the device, making it more suitable for analysis using current low sensitivity, high noise webcams or mobile phone cameras. In addition, since the images are taken at low speed, the file size is reduced by a factor of 40 [94].

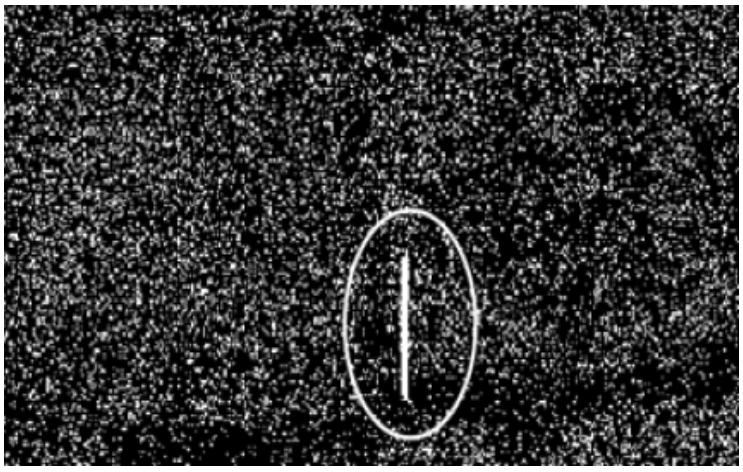


Figure 2.6: Image of a cell crossing a frame in streak mode cytometry.

One disadvantage of streak photography is that in the presence of many particles, their paths overlap too often impeding the separation of individual particles[99]. However, this is not a problem in the case of identifying CTCs since as was mentioned before, CTCs are in an extremely low concentration in blood.

Chapter 3: Algorithms Useful for Cell Tracking

3.1 Introduction

Cell detection and tracking are important problems with numerous clinical applications, and subjects of intense study in recent years. Despite inherent value, the tasks are too burdensome for manual methods, and would appear to be ripe for automation. However, several challenges confront automated detection and tracking, including poor image quality (e.g., low contrast, high noise levels), weak signals, and overlapping cells [100]. Cell tracking methods generally consist of two main image processing steps: (i) cell segmentation (the spatial aspect of tracking) that requires recognition and detection of potentially relevant objects (cells), including discrimination of these cells from the background. Segmentation allows the identification of objects inside an image [101], but provides little information about the relation of the objects with each other. This issue is addressed by the second major step for cell tracking, (ii) association of the relevant objects with each other across the whole image. This step is considered the temporal aspect of cell tracking[100].

3.2 Recognition of relevant objects or detection phase: Cell segmentation

Image segmentation is an important problem in digital imaging processing. Several methods useful for a wide range of applications have been reported for different fields of study [102,103] including cell recognition for clinical applications such as classification

of white blood cells, identification of bone marrow cells in images, detection of cancer cells in volumetric images of blood for leukemia diagnosis among others [104-111]. This discussion will focus on methods relevant to cell segmentation.

Segmentation is a process of partitioning a digital image into several parts or segments and extracting regions of interest (ROI) from these image segments [112]. Segments correspond to a set of pixels within a determined range, and the collection of all the segments represents the entire image. Segmentation can be described as a pixel labeling problem where a pixel is assigned to a class and each class is used to discriminate a significant structure in the image. The goal of segmentation is to discriminate the most meaningful areas of the image [100,113-115]. Segmentation methods are broadly classified as classical (based on filtering and statistical techniques) and non-classical approaches such as neural networks, fuzzy logic and genetic algorithms[116]. Classical segmentation approaches include three major categories (Figure 3.1): (i) Intensity-based (thresholding), (ii) Edge-based segmentation and (iii) Region-based segmentation [113,117]. Some literature describes other categories of segmentation methods such as Clustering-based methods, Watershed segmentation, etc. [112,117].

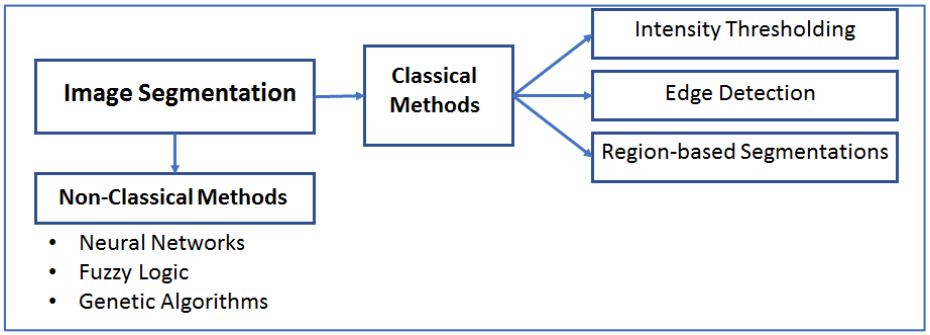


Figure 3.1: Classification of image segmentation methods. *Two major methods for classical image segmentation are thresholding and edge detection.*

3.2.1 Segmentation by intensity thresholding

Intensity thresholding is perhaps the simplest and reasonably technique to separate objects from the background. Thresholding has been extensively used for multiple applications in a variety of fields such as cell imaging, knowledge representation, ultrasonic image analysis, thermal imaging, computed tomography, confocal microscopy, etc. [118-126] and it constitutes an important tool for cell and particle tracking [127,128].

The fundamental principle of thresholding is based on a characteristic feature of the image such as pixel values that can be classified into two or more dominant groups [129,130]. These groups can be separated by a threshold T such that any point $p(x, y) > T$ in the threshold image $g(x, y)$ is called an object point. A point otherwise is called a background point. Thus the threshold $g(x, y)$ can be defined as follows[118,130]:

$$g(x,y) = \begin{cases} 1 \; if \; p(x,y) > T \\ 0 \; if \; p(x,y) \leq T \end{cases}$$

The threshold can be set manually (choosing a defined value as the threshold) or derived automatically from information based on an intensity histogram. Thresholding techniques can be further classified as local or global thresholding (Figure 3.2). In addition, global thresholding can be subclassified as Traditional, Iterative or Multistage thresholding [118,131]. Multistage thresholding is a two-stage method, with the first stage image divided into three sub-images: foreground and background and an additional

fuzzy image. In the second stage, the pixels are classified as belonging to one of the three images, with the pixels classified as belonging to the fuzzy image required to provide additional information to determine if each belongs to background or foreground. Multistage thresholding is used mostly for handwriting images Traditional thresholding (Otsu method) is the most widely used global thresholding method. Otsu method divides the image into two classes, foreground and background. Iterative methods (Triclass) are based on Otsu's method, but produce an additional class. Since these methods were more relevant in the development of the streak detection algorithm, they are discussed in more detail below. [118,131].

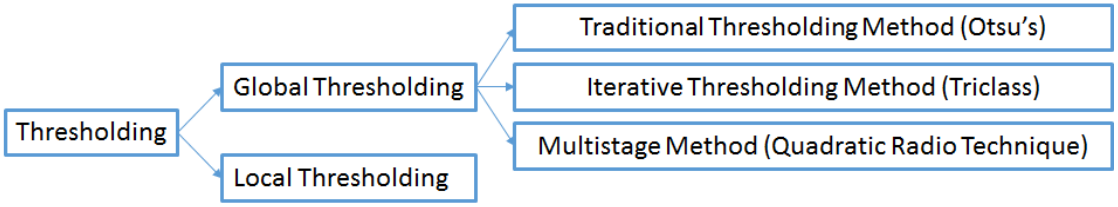


Figure 3.2: Classification of Thresholding *(figure adapted from [118]).*

Otsu method

Otsu's method, named after Nobuyuki Otsu, is a non-parametric unsupervised method for automatic threshold selection based on the variance of pixel intensity. The Otsu algorithm performs an exhaustive search for the optimal threshold by iterating through all the possible threshold values and calculating the spread of pixel intensity levels on both side of the threshold. The aim of Otsu's algorithm is to find the minimal intra-group variance (within-class variance). The within-class variance σ_W^2 is defined as a weighted sum of the background and foreground variance [118,129,131-133] as shown below:

$$\sigma_W^2 = \omega_b \sigma_b^2 + \omega_f \sigma_f^2$$

$\sigma_W^2 = Within\ Class\ Varianza;\ \omega = Weight; b = background; f = foreground$ [133]

A 24x14 pixels window corresponding to a section of a streak in one frame (Figure 3.3) will serve to illustrate the functionality of Otsu's algorithm (Figure 3.3).

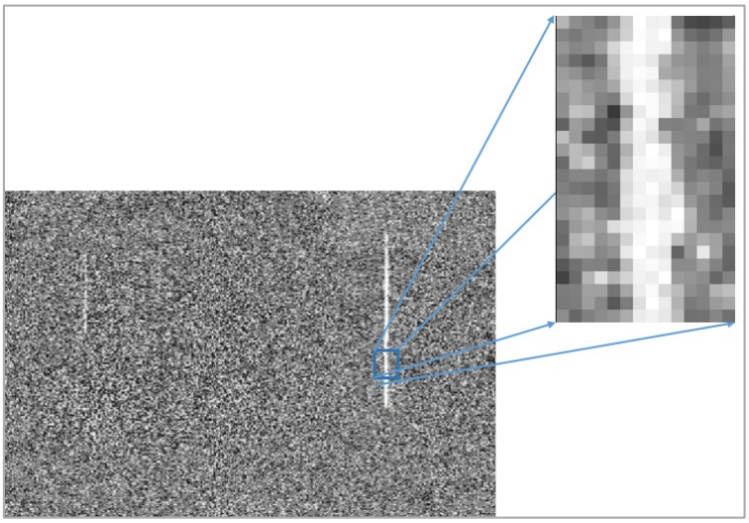
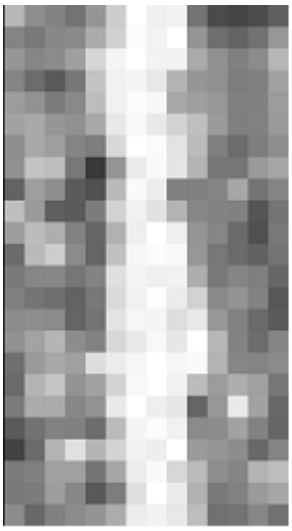
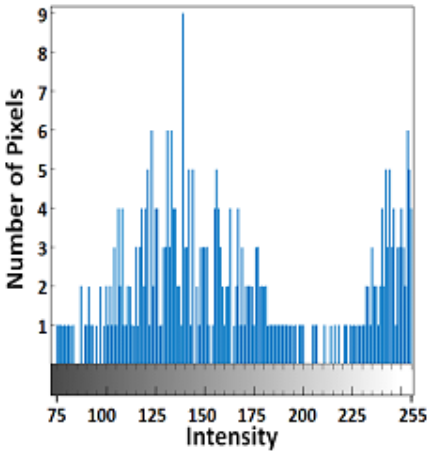


Figure 3.3: Amplified section of a streak. *Amplified 24x14 window, showing the streak (signal) and background.*

A histogram of the 24x14 window in figure 3.3 shows that the pixels of the image can be classified in two groups according to their intensity (Figure 3.4). The purpose of the Otsu algorithm is to find the optimal cut off or threshold that separates these groups (background from foreground).





Intensity	Number of Pixels
0	0
1	0
74	0
75	1
76	1
181	2
182	1
183	1
184	1
185	1
186	1
187	1
253	6
254	5
255	4

a *b* *c*

Figure 3.4: Histogram representation of a streak. *(a) amplified 24x14 window (a) 336 pixels section showing the streak and the background, (b) histogram of the window section. (c) table showing the number of pixels for each level of intensity.*

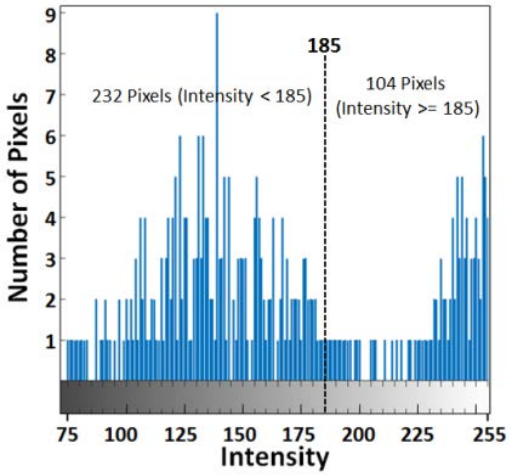
Otsu's method calculates the weight of the pixel in the background and the foreground for each possible threshold value from 0 to 255. The weigh is calculated by adding the number of pixels in the group (foreground or background) and dividing it by the total number of pixels as shown below.

$$Weight\ of\ Foreground \quad \omega_f = \frac{\text{Number of Foreground Pixels}}{\text{Total Number of Pixels}}$$

$$Weight\ of\ Background \quad \omega_b = \frac{\text{Number of Background Pixels}}{\text{Total Number of Pixels}}$$

To illustrate this effect, a threshold of 185 will be used as an example. Figure 3.5 shows the threshold value that separates the two groups. Pixels with intensity less than 185

are called background, the rest are called foreground. The weight of the background (noise) and the foreground (signal) is calculated below (Figure 3.5).



$$Weight \quad \omega_b = \frac{232}{336} = \mathbf{0.6905}$$

$$Weight \quad \omega_f = \frac{104}{336} = \mathbf{0.3095}$$

Figure 3.5: Otsu method weight calculations*. Histogram representation of the pixels from the24x14 window(right) and weight calculation (left).*

The next step is to calculate the variance for the background and foreground class. The variance is calculated by the sum of the squares of intensity minus the mean divided by the total number of pixels. The mean μ is the sum of each individual pixel intensity multiplied by the number of pixels with that intensity and divided by the total number of pixels in the class.

$$Mean\ \mu\ = \frac{SUM(Intesity * Number\ of\ pixels)}{Total\ Number\ of\ Pixels}$$

$$Variance\ \sigma^2\ = \frac{SUM(Intesity - \mu)}{Total\ Number\ of\ Pixels}$$

This process is illustrated below using the values of the table in figure 3.4. Pixels with intensity from 75 to 184 were used to calculate the background variance (there were no pixels with values of intensity lower than 75), and the rest of the pixels are used to calculate the foreground variance (pixels with intensity from 185 to 255)

Background

$$Mean \quad \mu_b = \frac{(75*1)+(76*2)+\ldots..+(183*1)+(184*1)}{232} = \frac{31390}{232} = 135.30$$

$$Variance \quad \sigma_b^2 = \frac{((75-135.30)^2*1)+\ldots+((184-135.30)^2*1)}{232} = \frac{160028.9}{232} = \mathbf{689.77}$$

Foreground

$$Mean \quad \mu_f = \frac{(185*1)+(186*1)+\ldots..+(254*5)+(255*4)}{104} = \frac{24277}{104} = 233.43$$

$$Variance \quad \sigma_b^2 = \frac{((185-233.43)^2*1)+\ldots.+((255-233.43)^2*4)}{104} = \frac{43299.43}{104} = \mathbf{416.34}$$

Within class variance at threshold value of 185

The last step of Otsu's algorithm is to calculate within class variance. For this example, the within class variance for threshold 185 is calculated as below:

$$\sigma_{185}^2 = 0.6905 * 689.77 + 0.3095 * 416.34 = \mathbf{605.14}$$

As was mentioned before the within class variance (intra-group variance) is calculated for all possible threshold values, and the optimal threshold is the one with the lowest intra-group variance.

It is important to notice that MATLAB function *"graytresh()"* computes a global threshold using Otsu's method, but normalizes the intensity to values between 0-1. The table in the figure below (Figure 3.6) shows the normalized values of intensity with the minimal within class variance at level 185.

Intensity	Number of Pixels	Within Class Variance(ω)	Normalized Intensity
0	0	2663.19	0
1	0	2663.19	0.0039
74	0	2663.19	0.2902
75	1	2663.19	0.2941
76	1	2638.65	0.2980
181	2	608.96	0.7098
182	1	606.64	0.7137
183	1	605.76	0.7176
184	1	605.26	0.7216
185	1	605.14	0.7255
186	1	605.41	0.7294
187	1	606.07	0.7333
253	6	2299.75	0.9922
254	5	2446.31	0.9961
255	4	2567.06	1

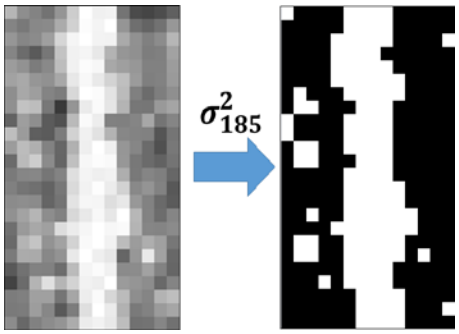


Figure 3.6: Within class variance and Otsu thresholding*. Right:The values of within class variance and norlized intesity for each level of intensity showingxx . Left: black&White image resulting from aplying Otsu method at intensity level of 185.*

Otsu's method is more effective in images with bimodal histograms, and the above example used an ideal area of the image where two distinctive peaks are easily identified, but in real-life applications where the peaks are not easily identified, Otsu's method can lead to suboptimal results[134]. Iterative methods have been reported to perform better than the traditional Otsu method for both synthetic and real images[134].

Iterative thresholding method (Triclass)

As mentioned before, the iterative method divides the pixels in three classes (Figure 3.7). In the first iteration, Otsu's method is applied as described above to obtain an

optimal threshold and the mean of the two classes (background and fore ground) is calculated. Based in the mean, the pixels in the histogram are divided into three classes. The pixels with values smaller that the small mean are classified as background; the pixels with values larger than the large mean are classified as foreground; and the pixels with values between the two classes (called "to-be-determined" or TBD) are used for the next iteration where the same procedure is applied again. The process continues until a pre-defined termination rule is reached, e.g., a pre-set threshold [118,131,134].

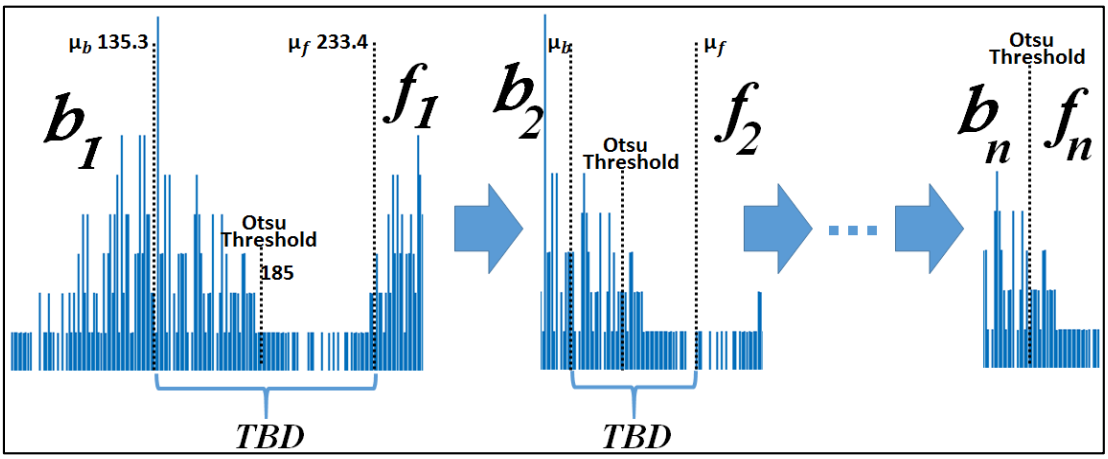


Figure 3.7: Representation of the Iterative Thresholding method. *In each iteration, the histogram becomes smaller until the termination rule is reached, and the optimal threshold is found.*

During the last iteration (determined by a preset condition) the remaining pixels are divided in only two classes. The final classes (background and foreground) are determined by the logical union for the classes from each iteration [118,131,134] as shown below.

$$\boldsymbol{Background} = \boldsymbol{b_1} \cup \boldsymbol{b_2} \cup \ldots \cup \boldsymbol{b_n}$$

$$\boldsymbol{Foreground} = \boldsymbol{f_1} \cup \boldsymbol{f_2} \cup \ldots \cup \boldsymbol{f_n}$$

3.2.2 Segmentation through edge detection

Edge detection is an alternative method to identify cells from the background. Edge is a basic characteristic feature of an image and is defined as the point at which an image sharply changes amplitude attributes such as brightness, luminescence or intensity, that give the observer the impression of an area transitioning between two different regions. Edge detection techniques take advantage of the discontinuity in brightness or intensity to identify the boundaries of the objects of interest. This boundary or edge represents points of significance changes in depth values determined by the local information of the pixels [113,135,136].

The changes in intensity can be found using first- or second-order derivatives. First order derivative-based edge detection operators (such as Roberts, Prewitt or Sobel operators) find edges by computing the image gradient. Second order derivative operators look into the second derivative's zero-crossing to find the edges [130,136].

First order derivative edge detection, gradient based algorithms

The image gradient is a measurement of rate of change in the image intensity. The gradient is defined as a vector oriented in the direction of the gradient, perpendicular to the edge and with the strength of the edge represented as the magnitude of the gradient vector. Since an image can be represented as a continuous derivative of the intensity of a group of pixels, then the image gradient can be calculated using the partial derivative at each pixel location [130,136].

The image gradient (G) is defined as:

$$G(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial y}{\partial x} \\ \frac{\partial y}{\partial x} \end{bmatrix}$$

where the image magnitude (M) is defined as: $M(G) = \sqrt{(g_x)^2 + (g_y)^2}$

The direction of the gradient is given by the angle (α) at points x, y:

$$\alpha(x, y) = \tan^{-1}\left[\frac{g_y}{g_x}\right]$$

Prewitt kernel

An approximation of the gradient of the image intensity function can be used for edge detection in digital images. For example, the Prewitt operator (Prewitt kernel, figure 3.8) convolves the image with a small 2-D (3x3) pixel mask to estimate the partial derivative of the image at a point in the x- and y- directions:

-1	-1	-1
0	0	0
1	1	1

(a)

-1	0	1
-1	0	1
-1	0	1

(b)

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

(c)

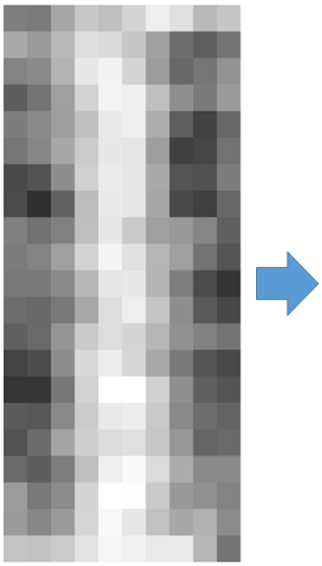
Figure 3.8: Prewitt mask and partial derivatives approximation*. a) 3x3 mask in x-direction. b) 3x3 mask in y-direction. c) Pixel location.*

An approximation of the partial derivatives can be found using the Prewitt operator as shown below:

$$g_x = \frac{\partial f}{\partial x} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + z_6 + z_9) - (z_1 + z_{42} + z_7)$$

To illustrate this procedure, the Prewitt operator will be used to calculate the image gradient in section of a frame containing a streak. In this example Prewitt's method will be implemented using convolution. Figure 3.9 shows a window section of a frame containing a streak with the intensity values represented in the matrix on the right of the image. Prewitt's method convolves the image with a 3x3 mask (Figure 3.8, yellow 3x3 window)). The mask is stepped across the image and with each step the central value (pixel z_5 in red) is replaced with the magnitude of the vector at that location. In this example the magnitude of the target pixel (in red, figure 3.9) will be calculated.



0.4863	0.5412	0.6275	0.7490	0.7529	0.8000	0.9255	0.8824	0.7725	0.7176
0.6118	0.6118	0.7020	0.8902	0.8863	0.7529	0.6353	0.4392	0.3569	0.4353
0.5176	0.5725	0.7451	0.8902	0.9373	0.8588	0.6157	0.3882	0.4275	0.5804
0.4157	0.3765	0.5843	0.8588	0.9569	0.9059	0.7529	0.6118	0.4980	0.5843
0.4745	0.5725	0.6235	0.7725	0.9176	0.8980	0.6784	0.3255	0.2745	0.4078
0.4510	0.5843	0.6314	0.7804	0.9490	0.8941	0.6235	0.2902	0.2745	0.4784
0.2902	0.3176	0.5255	0.7725	0.9294	0.9294	0.6510	0.3137	0.2980	0.5098
0.2824	0.1843	0.4039	0.7882	0.9098	0.8902	0.6353	0.3137	0.2549	0.4078
0.4863	0.4392	0.5059	0.7608	0.8902	0.7922	0.6196	0.6157	0.5765	0.3373
0.4902	0.5059	0.6039	0.8235	0.9529	0.8980	0.7137	0.5922	0.4235	0.3294
0.5098	0.4745	0.5412	0.7020	0.8471	0.9373	0.7843	0.4706	0.2902	0.2118
0.3961	0.4078	0.4824	0.6863	0.8784	0.8941	0.7569	0.5569	0.3569	0.2941
0.3882	0.4196	0.5529	0.7961	0.9137	0.7961	0.6745	0.5608	0.4745	0.4510
0.2588	0.2824	0.5373	0.8588	0.9490	0.8471	0.7373	0.4941	0.3686	0.2941
0.2235	0.2196	0.4784	0.8314	0.9725	0.9373	0.8392	0.5255	0.3294	0.3294
0.3412	0.3608	0.5333	0.7804	0.9294	0.9333	0.7686	0.5608	0.4431	0.4157
0.3294	0.4392	0.6235	0.7882	0.8863	0.8902	0.7608	0.5922	0.4157	0.3922
0.4157	0.3608	0.5059	0.7765	0.9412	0.9686	0.8510	0.6667	0.5333	0.5333
0.6000	0.4627	0.5961	0.8627	0.9647	0.9686	0.8431	0.6000	0.5765	0.5333
0.6392	0.5059	0.5922	0.8431	0.9647	0.9373	0.8118	0.6549	0.6824	0.5176
0.7529	0.7882	0.8118	0.8745	0.9373	0.9098	0.8627	0.8824	0.7725	0.4314

(a) *(b)*

Figure 3.9: Streak representation as pixel intensity values*. a) Section of a frame with a streak. b) pixel intensity represented as double.*

1) Each pixel is multiplied by the value in the corresponding location in the 3x3 mask (either -1, 1 or zero) as shows in figure 3.10. To ensure that the dynamic range of the output matches the input, in this case, the result is normalized by dividing by 6 e.g., for

the Prewitt operator, the sum of the absolute value of the mask is: |-1| + |-1| +|-1| + 1 + 1 +1 = 6.

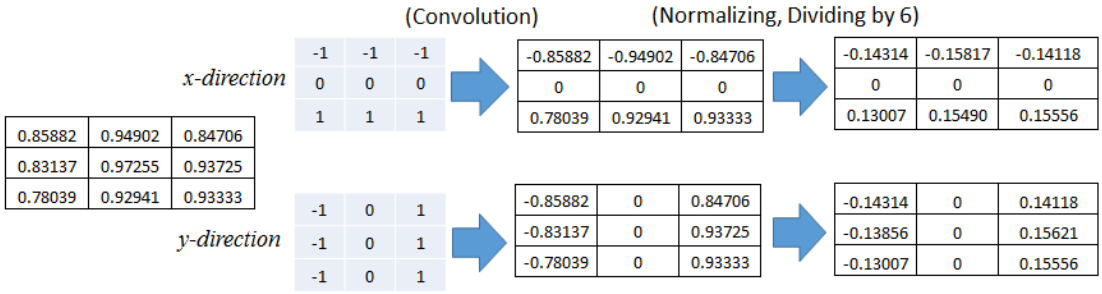


Figure 3.10: Convolution using a Prewitt mask. *Each pixel is multiplied by a corresponding value in the mask and divided by 6 to normalize it.*

2) The partial derivatives in *x*- and *y*-directions are estimated by adding all the values of the resulting matrix as indicated below:

$$g_x = \frac{\partial f}{\partial x} = (0.1300) + (0.1549) + (0.1555) + (-0.1431) + (-0.1581) + (-0.1411)$$

$$= -0.00196$$

$$g_y = \frac{\partial f}{\partial y} = (0.1411) + (0.1562) + (0.1555) + (-0.1431) + (-0.1385) + (-0.1300)$$

$$= 0.04118$$

3) The magnitude of the gradient is calculated using the formula below, and the target pixel is replaced with the value of the magnitude (Figure 3.11). The procedure is repeated until all the values of the pixels in the image are replaced by their corresponding magnitude value of their gradient.

$$M(G) = \sqrt{(-0.00196)^2 + (0.04118)^2} = 0.041223$$

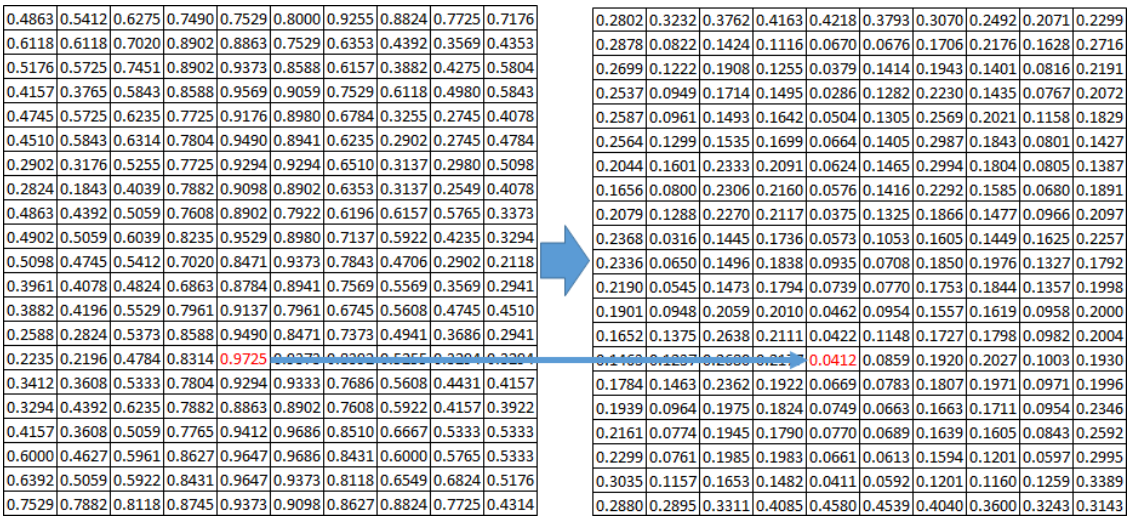
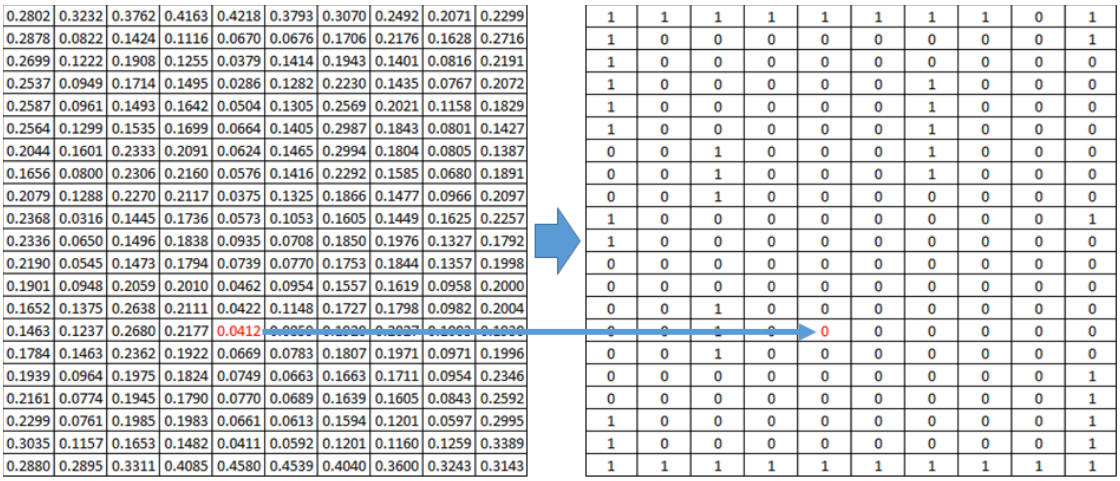


Figure 3.11: Gradient magnitude and pixels*. The target pixel is replaced with the magnitude of the gradient.*

4) Using a predetermined threshold value (for this example the threshold used is 0.18) the image is binarized assigning '1' to the edges and '0' to the background as shown below:

$$g(x,y) = \begin{cases} 1 \; if \; p(x,y) > Threshold \\ 0 \; if \; p(x,y) \leq Threshold \end{cases}$$

Using this procedure, all pixels with magnitude values under 0.18 are suppressed (set to zero), and all the other pixels are set to 1. Pixels set to '1' are considered to be the edges of the object in the image. The result of this operation is illustrated in figure 3.12, in this case the target pixel in this example (in red) ended up being a non-edge pixel.



0.2802	0.3232	0.3762	0.4163	0.4218	0.3793	0.3070	0.2492	0.2071	0.2299
0.2878	0.0822	0.1424	0.1116	0.0670	0.0676	0.1706	0.2176	0.1628	0.2716
0.2699	0.1222	0.1908	0.1255	0.0379	0.1414	0.1943	0.1401	0.0816	0.2191
0.2537	0.0949	0.1714	0.1495	0.0286	0.1282	0.2230	0.1435	0.0767	0.2072
0.2587	0.0961	0.1493	0.1642	0.0504	0.1305	0.2569	0.2021	0.1158	0.1829
0.2564	0.1299	0.1535	0.1699	0.0664	0.1405	0.2987	0.1843	0.0801	0.1427
0.2044	0.1601	0.2333	0.2091	0.0624	0.1465	0.2994	0.1804	0.0805	0.1387
0.1656	0.0800	0.2306	0.2160	0.0576	0.1416	0.2292	0.1585	0.0680	0.1891
0.2079	0.1288	0.2270	0.2117	0.0375	0.1325	0.1866	0.1477	0.0966	0.2097
0.2368	0.0316	0.1445	0.1736	0.0573	0.1053	0.1605	0.1449	0.1625	0.2257
0.2336	0.0650	0.1496	0.1838	0.0935	0.0708	0.1850	0.1976	0.1327	0.1792
0.2190	0.0545	0.1473	0.1794	0.0739	0.0770	0.1753	0.1844	0.1357	0.1998
0.1901	0.0948	0.2059	0.2010	0.0462	0.0954	0.1557	0.1619	0.0958	0.2000
0.1652	0.1375	0.2638	0.2111	0.0422	0.1148	0.1727	0.1798	0.0982	0.2004
0.1463	0.1237	0.2680	0.2177	0.0412	0.0859	0.1639	0.2027	0.1093	0.1639
0.1784	0.1463	0.2362	0.1922	0.0669	0.0783	0.1807	0.1971	0.0971	0.1996
0.1939	0.0964	0.1975	0.1824	0.0749	0.0663	0.1663	0.1711	0.0954	0.2346
0.2161	0.0774	0.1945	0.1790	0.0770	0.0689	0.1639	0.1605	0.0843	0.2592
0.2299	0.0761	0.1985	0.1983	0.0661	0.0613	0.1594	0.1201	0.0597	0.2995
0.3035	0.1157	0.1653	0.1482	0.0411	0.0592	0.1201	0.1160	0.1259	0.3389
0.2880	0.2895	0.3311	0.4085	0.4580	0.4539	0.4040	0.3600	0.3243	0.3143

1	1	1	1	1	1	1	1	0	1
1	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	1	0	0	0
0	0	1	0	0	0	1	0	0	0
0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	1
1	1	1	1	1	1	1	1	1	1

***Figure 3.12: Binary image representation**. The image from the previous step is binarized (threshold = 0.18)*

MATLAB implementation of edge detection is done using the function *"edge()"*. The MATLAB command *"PrewittImage = edge(StreakImage, 'Prewitt', 0.18)"* return the resulting filtered image using Prewitt's operator with a threshold of 0.18. As is shown in figure 3.13, the convolution method returns the same image as the MATLAB *"edge()"* function. Figure 3.13 shows that Prewitt method successfully identify the streak's edges on the right of the frame but missed the edges of the streak on the left.

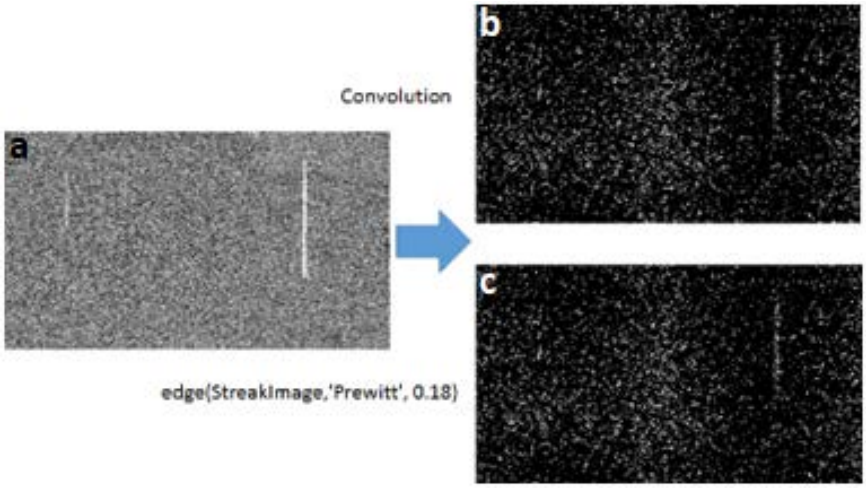


***Figure 3.13: Edge detection using the Prewitt operator**. a) Original image. b) Thresholding using convolution. c) Thresholding using MATLAB function "edge()".*

Some issues with gradient based methods are their lack of flexibility to modify the kernel or coefficients, and they are very sensitive to noise. Second order derivative methods can minimize these issues [136].

Second order derivative, zero crossing edge detection method

The magnitude of the first derivative allows the observer to infer the presence of an edge. For accurate detection of boundaries, a single pixel-wide boundary is desirable if the edge is discrete, but this is challenging for gradient-based methods since these search for the maximum derivative. An alternative to the maximum derivative is the zero-crossing effect of the second derivative that is the intersection between the zero intensity and the extrema of the second derivative (usually the zero-crossings of the Laplacian image). The advantage of the zero-crossing method is that it results in a single pixel-wide boundary [130,137].

The sign of the second derivative determines if an edge pixel is in the dark side or the bright side of the edge, and the point where the Laplacian changes sign, is the zero crossing [130,137,138]. Laplacians can be calculated by summing the second partial derivative in the *'x'* direction and the second partial derivative in the *'y'* direction. The formula for the Laplacian *L(x, y)* is as follows:

$$L(x,y) = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y}$$

A discrete approximation of the second partial derivative for the Laplacian operator can be found as follows:

$$'x'\ direction: \quad \frac{\partial^2 f}{\partial^2 x} = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

$$'y'direction: \quad \frac{\partial^2 f}{\partial^2 x} = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

To illustrate this procedure a horizontal section of a streak (1x13 pixels) will be used as example (Figure 3.14).

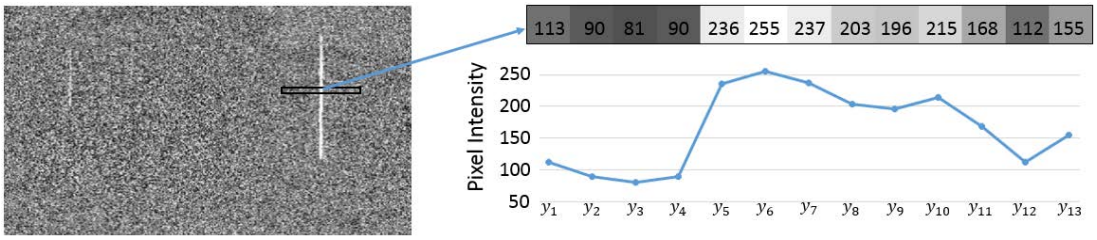


Figure 3.14: Graphic representation of a 1x13 pixel windows showing the pixels intensity values.

In this example, the second partial derivative for pixel y_6 as a function of the intensity in the 'y' direction is calculated as follows:

$$\frac{\partial^2 f}{\partial^2 x} = f(x,y_6+1) + f(x,y_6-1) - 2f(x,y_6)$$

$$= f(x,y_7) + f(x,y_5) - 2f(x,y_6)$$

Since *'x'* is constant when moving in *'y'* direction, *'x'* values can be neglected:

$$\frac{\partial^2 f}{\partial^2 x} = f(y_7) + f(y_5) - 2f(y_6)$$
$$= (237) + (236) - 2(255)$$
$$= (237) + (236) - (510)$$
$$= -37$$

Using the same procedure, the second derivative for all pixels in the horizontal section (1x13 pixels) is calculated and values plotted to identify the zero crossing as shown in figure 3.15. As mentioned above, the sign of the partial derivative indicates which side of the edge the pixels are located. For the brighter side of the edge, the partial derivative has a negative sign and a positive sign for the darker side of the edge, therefore the zero crossing (when the partial derivative changes sign) indicates the boundary of change in intensity. Figure 3.15 shows a marked change of intensity from pixel y_4 to pixel y_5 (from 90 to 336) where the edge of the streak is located. These changes in intensity are detected by the change in the sign of the second derivative (from +137 to -127). From pixel y_5 the intensity increased until pixel y_7, which changed the sign of the second derivative in pixel y_8 (+27). Between pixel y_9 and pixel y_{10} another change in intensity from 196 to 216 is detected (the derivative changed from +26 to -66). In pixel y_{12}, the intensity increases to 112 changing the sign of the derivative again (from-9 to +99).

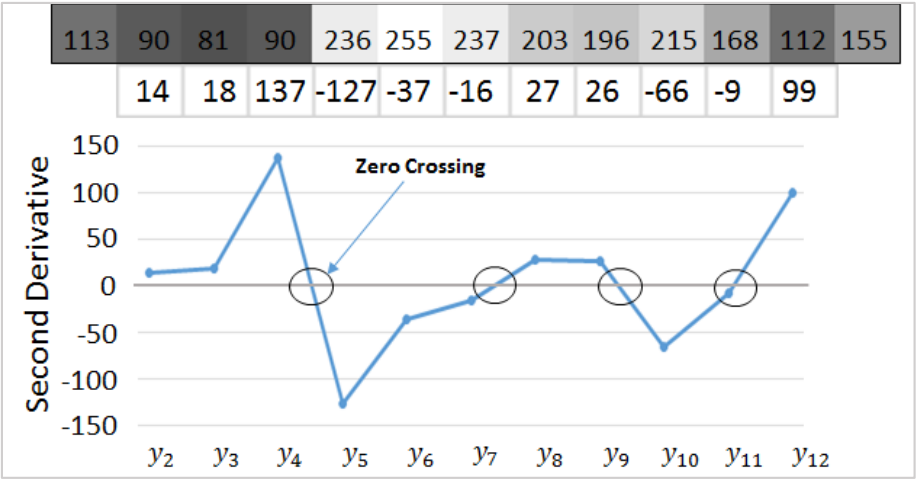


Figure 3.15: Second partial derivative and zero crossing points.

Note that the zero crossing between pixels y_4 and y_5 in figure 3.15, indicates a true edge of the streak, but the zero crossing between pixels y_{10} and y_{11} might not represent a true edge and probably correspond to noise.

The Laplacian of the image can be calculated adding both partial derivatives, as showed below.

$$L(x,y) = [f(x+1,y) + f(x-1,y) - 2f(x,y)] + [f(x,y+1) + f(x,y-1) - 2f(x,y)]$$

$$= f(x+1,y) + f(x-1,y) - 4f(x,y) + f(x,y+1) + f(x,y-1)$$

$$= 1 * f(x+1,y) + 1 * f(x-1,y) - 4 * f(x,y) + 1 * f(x,y+1) + 1 * f(x,y-1)$$

This corresponds to the Laplacian operator for digital images and can be represented in the Laplacian kernel (Figure 3.16):

0	1*f(x-1,y)	0		0	1	0
1*f(x,y-1)	-4*f(x,y)	1*f(x,y+1)	→	1	-4	1
0	1*f(x+1,y)	0		0	1	0

Figure 3.16: Laplacian kernel*. The operator can be calculated adding the second partial derivatives (x and y directions) of the image.*

To illustrate the results of this process the Laplacian kernel (combined with the MATLAB *conv2()* function) was used to identify the edges of the streak in the image in figure 3.17. The Laplacian kernel was able to identify the two streaks in the frame, but also picked up some noise.

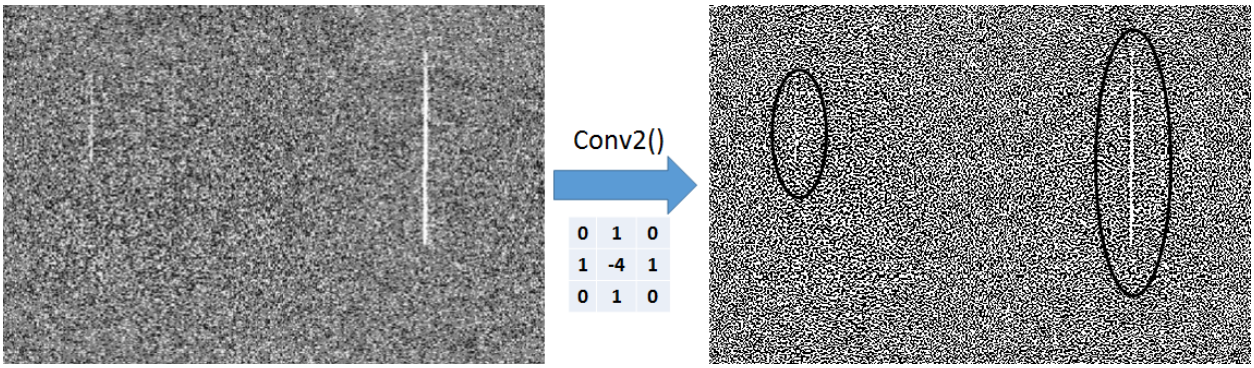


Figure 3.17: Applying Laplacian operator to streak imagine.

The above example shows one of the limitations of the second derivative: high sensitivity to noise. To mitigate this issue, a Gaussian filter (see formula below) may be used to filter out noise as the first step in second derivative-based methods.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\pi\sigma^2}}$$

Combining the Laplacian and Gaussian filter equations yields the Laplacian of Gaussian *(LoG)* operator (the Marr-Hildreth method) for edge detection:

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1-\frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Figure 3.18 shows the LoG operation using the MATLAB function *"edge()"*, with a threshold of 0.017. The LoG operation eliminates noise, but has the expense of losing signal, as seen in figure 3.18, it produces fragmented edges. LoG has another potential problem: it can generate false edges. These issues require a more nearly optimal operator:

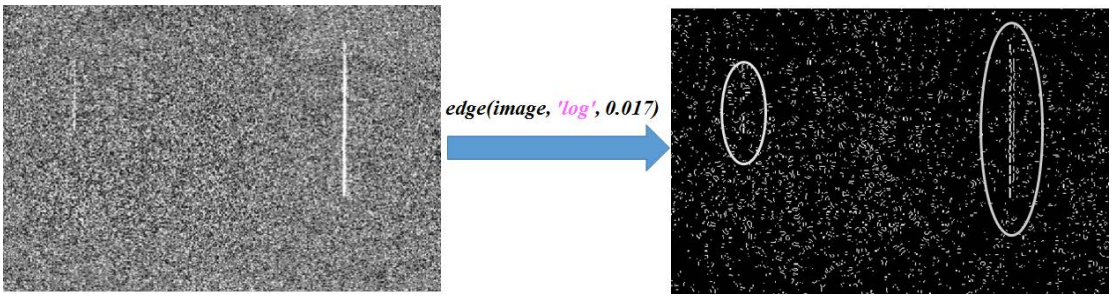


Figure 3.18: LoG operator*. The LoG operator applied to streak images can cause fragmentation.*

Canny edge detector operator

The Canny edge detection method was developed by John F. Canny in 1986[139] with the idea that a more nearly optimal operator could be derived to find edges in the presence of noise. The approximation of a more nearly optimal edge detector is found in the first

derivative of the Gaussian image in the direction of the gradient. The Canny method has since been extensible used in many applications, and is discussed in more detail below.

According to Canny, a more nearly optimal edge detector operator must have the following characteristics:

- Minimize the probability of false edge production and missing real edges.
- Detected edges must be close to true edges.
- Provide only one point for each true edge point.

Canny is a multistep algorithm that requires application of the Gaussian filter to smooth the image before starting the edge identification process. After the Gaussian filter is applied to the image, the process continues with the following steps: (i) Find the gradient direction of each pixel and quantize in one of 4 directions, (ii) Apply a non-maximum suppression method to provide only one point for each edge pixel, and (iii) Eliminate weak edges, these steps are described in more detail below.

Step 1: Finding the gradient direction and magnitude

After smoothing the image using Gaussian filtering, next calculate the gradients of the image in both directions. As we have seen earlier, several operators could be used to find the magnitude and direction of the gradient. Canny-based methods usually use the Sobel operator (Figure 3.19) to calculate the direction of the gradient (where the edge is perpendicular to the gradient).

-1	-2	-1
0	0	0
1	2	1

a

-1	0	1
-2	0	2
-1	0	1

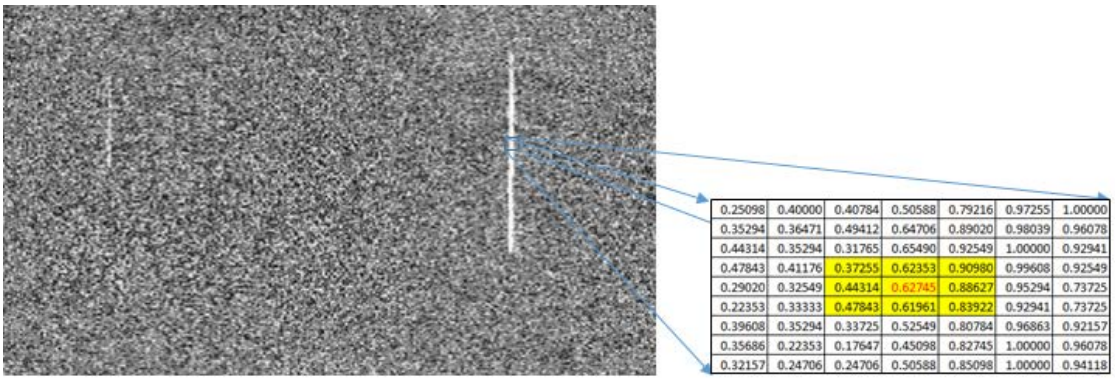
b

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

c

Figure 3.19: Sobel kernel*. a) 3x3 mask in x-direction. b) 3x3 mask in y-direction. c) Pixel location.*

To illustrate this process, the magnitude and the direction of the gradient using Sobel operator over a streak image will be calculated as (section in yellow, figure 3.20)



0.25098	0.40000	0.40784	0.50588	0.79216	0.97255	1.00000
0.35294	0.36471	0.49412	0.64706	0.89020	0.98039	0.96078
0.44314	0.35294	0.31765	0.65490	0.92549	1.00000	0.92941
0.47843	0.41176	0.37255	0.62353	0.90980	0.99608	0.92549
0.29020	0.32549	0.44314	0.62745	0.88627	0.95294	0.73725
0.22353	0.33333	0.47843	0.61961	0.83922	0.92941	0.73725
0.39608	0.35294	0.33725	0.52549	0.80784	0.96863	0.92157
0.35686	0.22353	0.17647	0.45098	0.82745	1.00000	0.96078
0.32157	0.24706	0.24706	0.50588	0.85098	1.00000	0.94118

Figure 3.20. Pixel values of a section of a streak*. A 9x17 pixel window from a section of a streak, with pixel values shown to the right.*

Calculating partial derivatives using Sobel operator

Based on the Sobel mask, the formulas to approximate the derivative of the gradient can be modified as follow:

$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2 * z_8 + z_9) - (z_1 + 2 * z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2 * z_6 + z_9) - (z_1 + 2 * z_{42} + z_7)$$

Applying a procedure similar to the Prewitt calculations, and using the modified formulas from Sobel kernel, the partial derivatives in *x* and *y* direction are calculated as showed in figure 3.21.

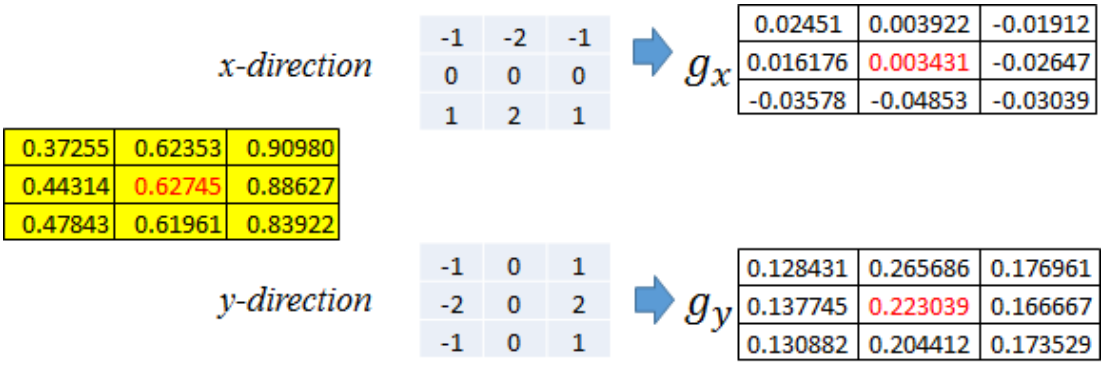


***Figure 3.21: Sobel operator**. Applying Sobel operator to the image and calculating the partial derivatives in x direction (g_x) and y direction (g_y). The result of the partial derivatives were normalized by dividing by 8.*

Calculating the edge angle at each pixel location

Using partial derivatives, the gradient and the edge angle can be calculated (the direction of the gradient is given by the angle (α) at points x, y).

$$\alpha(x,y) = \tan^{-1}\left[\frac{g_y}{g_x}\right]$$

$$\alpha(x,y) = \tan^{-1}\left[\frac{0.223039}{0.003431}\right]$$

$$\alpha(x,y) = tan^{-1}\,[65.00]$$

$$\alpha(x,y) = 1.555413$$

Converting $\alpha(x,y)$ to degrees:

$$\alpha(x,y)_{degrees} = 89.1186°$$

The angle in degrees is replaced in the position of each pixel as showed in figure 3.22

0.25098	0.40000	0.40784	0.50588	0.79216	0.97255	1.00000
0.35294	0.36471	0.49412	0.64706	0.89020	0.98039	0.96078
0.44314	0.35294	0.31765	0.65490	0.92549	1.00000	0.92941
0.47843	0.41176	0.37255	0.62353	0.90980	0.99608	0.92549
0.29020	0.32549	0.44314	0.62745	0.88627	0.95294	0.73725
0.22353	0.33333	0.47843	0.61961	0.83922	0.92941	0.73725
0.39608	0.35294	0.33725	0.52549	0.80784	0.96863	0.92157
0.35686	0.22353	0.17647	0.45098	0.82745	1.00000	0.96078
0.32157	0.24706	0.24706	0.50588	0.85098	1.00000	0.94118

87.5104	-80.426	-80.34	88.9132	85.4095	86.4677	-83.784
-17.865	88.5679	-85.389	79.1752	73.3147	71.5651	63.2394
-44.226	-65.556	-78.69	-86.033	88.7127	80.5377	76.3903
30.2564	65.9245	79.1956	89.1544	-83.834	19.2593	57.4772
16.007	-56.083	83.302	89.1186	-80.975	44.4213	66.0646
-29.745	84.8056	-74.709	-76.645	-80.066	-60.154	-69.748
-31.43	6.34019	-44.226	-71.565	-86.015	36.2538	-60.43
48.3178	52.6961	-70.959	-87.921	87.2422	75.0686	-86.315
-26.378	-31.04	48.9182	76.5116	84.3595	-86.76	86.9872

Figure 3.22: Edge angle*. The pixel values are replaced by angle in each pixel location.*

The Canny algorithm allows the edge direction to have one of four possible orientations: Horizontal edge (0°), Vertical edge (90°) and two diagonal edges (45° and 125°). To calculate the edge direction, Canny approximates the angle value to those orientations as follows:

1) If the angle is between 0° and 22.5° or between 180° and 157.5°, the angle is approximated to 0°, and the edge is a horizontal edge (yellow area in figure 3.23).

2) If the angle is > 22.5° and < 67.5°, the angle is approximated to 45°, and the edge is considered to be a diagonal edge (pink area in figure 3.23).

3) If the angle is > 67.5° and < 112.5°, the angle is approximated to 90°, and the edge is considered to be a vertical edge (green area in figure 3.23)

4) If the angle is > 112.5° and < 157.5° the angle is approximated to 135°, and the edge is considered to be a diagonal edge (blue area in figure 3.23).

*Similarly, this principle is applied for negative angles as show in figure 3.23

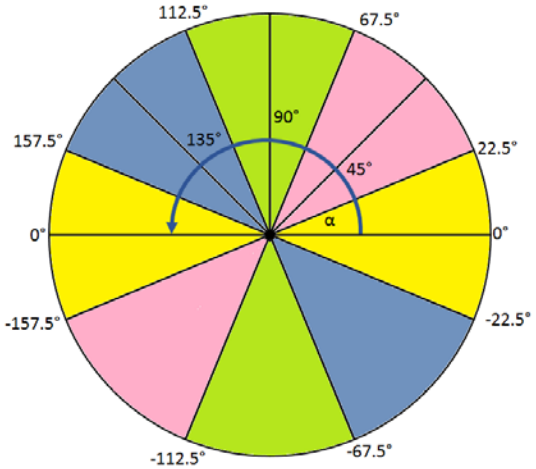


Figure 3.23: Calculating edge angles (Canny method). *Graphic representation for approximating angles used in Canny edge detection. The four possible orientations are show in yellow (horizontal), pink and blue (diagonal) and green(vertical).*

Using the above scheme (Figure 3.23), the angle values are approximated for each pixel, see below (Figure 3.24).

87.5104	-80.426	-80.34	88.9132	85.4095	86.4677	-83.784
-17.865	88.5679	-85.389	79.1752	73.3147	71.5651	63.2394
-44.226	-65.556	-78.69	-86.033	88.7127	80.5377	76.3903
30.2564	65.9245	79.1956	89.1544	-83.834	19.2593	57.4772
16.607	-56.083	83.302	89.1186	-80.375	44.4213	66.0646
-29.745	84.8056	-74.709	-76.645	-80.066	-60.154	-69.748
-31.43	6.34019	-44.226	-71.565	-86.015	36.2538	-60.43
48.3178	52.6961	-70.959	-87.921	87.2422	75.0686	-86.315
-26.378	-31.04	48.9182	76.5116	84.3595	-86.76	86.9872

90	90	90	90	90	90	90
0	90	90	90	90	90	45
135	135	90	90	90	90	90
45	45	90	90	90	0	45
0	135	90	90	90	45	45
135	90	90	90	90	135	90
135	0	135	90	90	45	135
45	45	90	90	90	90	90
135	135	45	90	90	90	90

Figure 3.24: Replacing edge angles in the figure matrix*. The angles are approximated to one of the 4 possible values and replaced in each pixel location.*

The edge direction and the gradient can be found using the angle values. Figure 3.25 shows examples of edge and gradient on different pixels. In this example the central pixel (in red) is considered a vertical edge with the gradient moving from left to right.

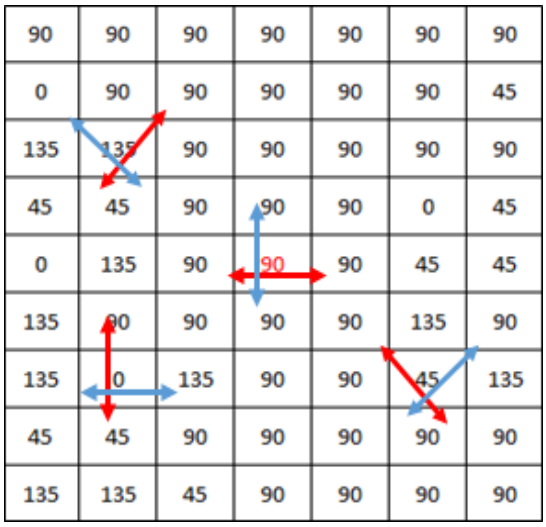


Figure 3.25: Gradient and edge direction in the figure matrix*. Red and blue arrows represent the gradient and edge direction respectively. Pixels with angle of 45^o or 135^o are considered diagonal edges, pixels with angle of 0^o are considered horizontal edges and pixels with angle of 45 or 135 are diagonal edges.*

Calculate the edge gradient magnitude at each pixel location

As was mentioned earlier, the magnitude of the gradient can be calculated using the partial derivative as follows:

$$\boldsymbol{M(G) = \sqrt{(g_x)^2 + (g_y)^2}}$$

Using the partial derivatives in both directions calculated in figure 3.21, the magnitude of the gradient, for this example, is found applying the above formula.

$$M(G) = \sqrt{(0.003431)^2 + (0.223039)^2} = 0.223066$$

Then, the magnitude values are replaced in each pixel location in the whole image as shown in figure 3.26. The magnitude and the orientation of the gradient is used in the next step to identify the edges.

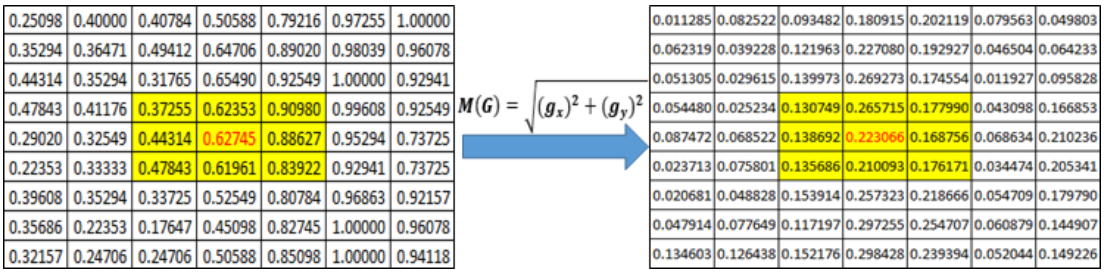


Figure 3.26: Gradient magnitude*. The magnitude of the gradient is replaced in each pixel.*

Step 2: Non-maximum suppression

Recalling the third requirement mentioned above for the Canny algorithm, the algorithm must provide only one point for every to true edge point. This requirement is achieved by using the gradient direction and magnitude of each pixel through the non-maximum suppression method. In this method, each pixel is compared with the neighbors at both sides of the gradient: if the magnitude of the pixel gradient is the largest compared to its neighbors, the pixel value is preserved. Otherwise the pixel is suppressed (set to zero). This process is illustrated in figure 3.27.

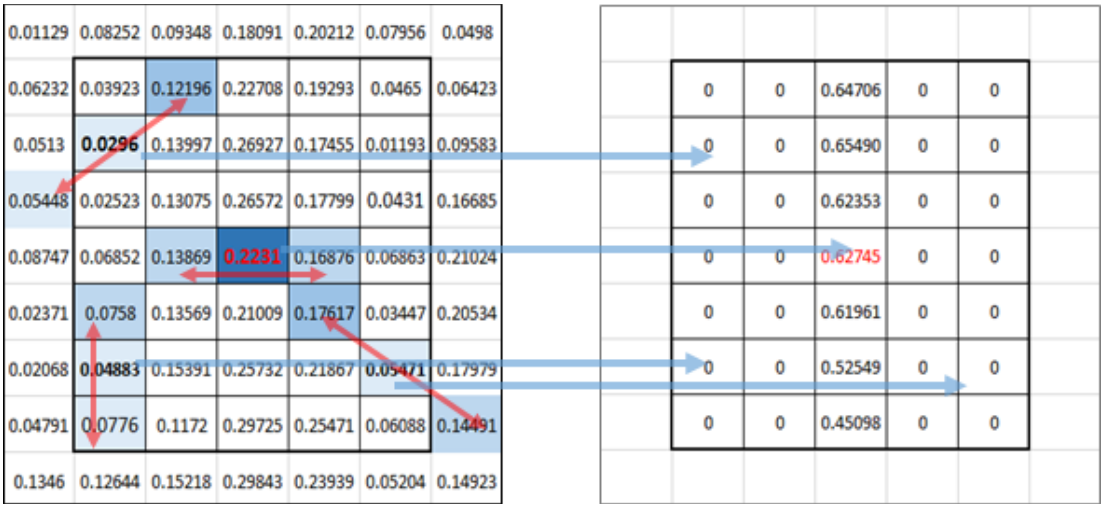


Figure 3.27: Gradient direction*. The red arrow represents the gradient direction. If the magnitude of a pixel is smaller than its neighbors on either side of the gradient (light blue color), the pixel is suppressed (set to zero). In this example, only the intensity values of the pixels in the 4th column were preserved, which identifies the vertical edge from the background.*

Step 3: Hysteresis thresholding

This is the last step in the Canny algorithm: The Hysteresis threshold is a double threshold applied to the image for two ranges of gray scale, such that, for an edge pixel in image I_p, with wide threshold τ_2 and narrow threshold τ_1, if the gradient value of pixel I_p is greater than the wide threshold, the pixel is considered a strong edge pixel. Otherwise, if the pixel value is lower than the wide threshold and larger than the narrow threshold, the pixel is considered a weak edge pixel. If the pixel value is smaller than the narrow threshold, then the pixel is suppressed (set to zero). Finally, all weak pixels not connected to a strong pixel are also suppressed [140,141].

Thus, the Hysteresis threshold H for pixel I_p can be defined as follow:

$$H(I_p) = \begin{cases} 1 & if\ I_p \geq \tau_2 \\ 1 & if\ I_p \leq \tau_2\ and\ I_p \geq \tau_1\ and\ I_p\ connected\ to\ a\ strong\ edge\ pixel \\ 0 & otherwise \end{cases}$$

To illustrate the importance of the threshold selection in the Canny method, figure 3.28 shows the result of applying the MATLAB function *"edge()"* with the Canny option to use the different high-low threshold combination. The high threshold is used for low edge sensitivity and low threshold is used for high edge sensitivity. The MATLAB "*edge()*" function can choose threshold values automatically depending of the characteristics of the input data (Figure 3.28.a,b). Increasing the threshold value results in less sensitivity in detection and therefore fewer detected edges, which results in the

elimination of unwanted edges (noise), but can also produce fragmentation of the edges or result in completely missing weak edges of foreground objects (Figure 3.28.c).

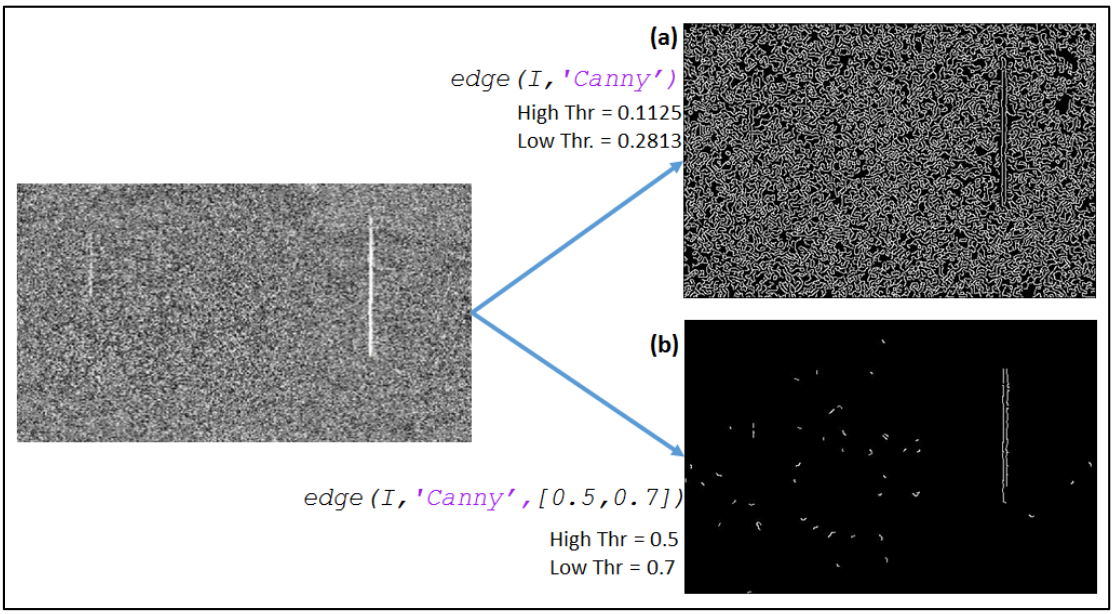


Figure 3.28: Canny method, threshold selection. *Using MATLAB function edge() with Canny option. (a) Threshold automatically selected by the system. (b) Increasing low and high threshold values result in the elimination of most of the unwanted edges (noise), but also cause fragmenting of the streaks, or completely missing the streaks.*

There are other methods for cell segmentation such as a template matching approach, watershed and others. In the template matching method, a template of the shape of the cell is created, matched, or otherwise is suppressed. This method works well for cells that do not change shape significantly but has limited value when significant variation in cell morphology is encountered [100]. Other methods apply region-based segmentations where objects and background are segmented by combining morphological filters and used as the base for watershed segmentation[142]. In watershed segmentation, the image is treated as topographic relief with an algorithm based on an

immersion analytical process. The image is considered much like a topographic map where the high is equivalent to the intensity subdividing the image into regions based delimited contours. The algorithm is sensitive to noise and tend to yield over-segmentation[100,143,144]

3.3 Association of the relevant objects with each other

After the objects are identified through segmentation, they can be associate and linked with other objects in different frames. For the purpose of streak detection and association, this can be achieved by associating cells with the nearest cells in the next frame using the centroid position. One problem with this approach is potential mismatch with a nearest not related cell. To avoid this problem other parameters can be considered for cell association such as: similarities in intensity, shape, surface area, orientation of major and minor axes, estimated displacement etc. A combination of these features will reduce the chance of mismatches [100]. Several methods for object tracking have been proposed for tracking of cells such as parametric active contours (snakes) or mean-shift based methods for tracking cancer cells and others[100,115,145,146]. These methods were designed to account for the complexity of the cell movement by considering different cell shape, size, speed, and direction. However, streak imaging has several advantages that facilitate the tracking of the cells. For example, the cells are represented by uniform shape (streaks) with comparable size moving in only one direction with very little variation from the straight line. In addition, due to the rare nature of the cells, they don't usually overlap, which allows the prediction of their position in consecutive frames.

These characteristics of streak cytometry imaging allows the use of simple methods such a cell association through centroid position for tracking the streaks across the frames.

3.4 Cell identification

Computational methods for cell detection in streak imaging (either through thresholding or edge detection) must be very sensitive due to the rare nature of the cells under analysis. High sensitivity comes with the side effect of an overwhelming number of spurious cells identified as cells, for example a sample with around 30 non-spurious cells can have hundreds of spurious cells. To address this issue an image classification method needs to be implemented.

Several methods including morphometric-based filters, or image feature methods such as 'bag of features' combined with machine learning classifiers can be applied to address this issue.

Bag of features (BoF)

The BoF approach represents an adaptive method to model image structures. In contrast to image segmentation, BoF looks for small characteristic regions that may represent the images. These small regions are extracted from all the images in the collection, and summarized in a set of code words or visual dictionary. The images can be represented as an orderless collection of local features without large scale spatial information and no relative location (bag) that correspond to the frequency of the code words that its contains[101]. BoF is derived from a bag-of-words model used in natural language

processing where a document can be represented as a normalized histogram of words count. A dictionary of words is extracted from the document and a term vector is created where each element in the vector is a word in the dictionary and the magnitude of the element is the number of times that the term appears in the document divided by the total number of words in the dictionary (the dictionary omits non-informative words and uses a single term for synonyms)[147].

Similar to a bag-of-words, a bag-of-features consists of visual vocabulary or codewords that correspond to features extracted from the set of images (for the BoF each feature cluster corresponds to a visual word) [147,148]. BoF has been successfully used in object detection, identification of visual patterns in biomedical images, image classification, etc. [101,147].

Chapter 4: Computational Tools for Cell Detection and Tracking

4.1 Introduction

Commercial and public domain software tools and methods for cell motion tracking have been intensively reviewed [149]. Some of those tools such as TimeLapseAnalyzer (TLA) [150] provide a framework for cell detection, but require a user's own implementation of an image pre-processing pipeline (e.g., thresholding, and object detection). General computational tools such as ImageJ [151] provide capabilities for development of cell tracking tools such as iTrack4U (an automatic cell tracking program based on a mean-shift algorithm) and ImageJ libraries [152] or MTrack2 [153] (an ImageJ plug in). MTrack2 relies on manual thresholding for streak detection and size filtering to eliminate small artifacts [135]. MTrack2 identifies fluorescence objects in each frame, and then determines which objects in successive frames are closest together. If these objects are within a user-defined distance (the maximum velocity of the objects) they are assembled into tracks. When multiple objects are within the distance determined by the maximum velocity the closest object is selected and the object is flagged in the output [135,149,153]. Some tools such as CellTrack [154], and CellTracker [155], perform image segmentation using classic edge detection algorithms (such as the Canny edge detector) for detecting cells.

As mentioned above, most of the available computational tools for cell detection and tracking use intensity thresholding or edge detection for segmentation. Two representative tools for these methods are: MTrack2 (thresholding-based) and

CellTrack (edge detection-based). These methods were used to compare the performance of the first streak detection algorithm described in the next section (Streak Detection Algorithm and Preliminary Results).

4.2 MTrack2

MTrack2 [153] is an ImageJ plugin originally developed at the University of California San Francisco in 2004 to track and analyze the velocity and directional persistence of movements of Caenorhabditis elegans (roundworm) [156]. MTrack2 is based on the MultiTracker plugin [157] (University of Texas Austin) which is based in the Object Tracker plugin [158]. MTrack2 identifies objects in each frame and uses the object's center of mass (centroid) to track direction and distance covered in subsequent frames. If the centroid location is within a user-specified distance (maximum velocity), MTrack2 groups the objects in tracks (each track can contain one or more objects). In the case that multiple objects are within a user-specified distance, the closest object is considered to belong to the same track.

For this dissertation, the objects to be identified by MTrack2 are individual streaks. Note that streaks are usually fragmented and MTrack2 identifies each fragment as a different object. Therefore, further processing using a MATLAB application was used to associate the objects that belong to the same streak in the same frame. Using the centroid location (provided by MTrack2) the objects detected by MTrack2 are grouped in streaks in different frames and the streaks are identified as cells.

MTrack2 allows the user to select the minimum track length (number of frames where the user expects to find the same object). For our purposes, due to their length, the streaks are expected to show in no more than four frames, but since MTrack2 can miss a streak in a frame, we set the minimum track length to one frame. MTrack2 also allows us to set the minimal object size for streak detection. For this work we set the minimum object size to 20 pixels to reduce the chance for detecting small artifacts [159]. Critical for the streak detections using MTrack2 is threshold selection. Manual threshold selection is done before running the MTrack2 to determine which objects will be detected and counted (Figure 4.1). For this dissertation we set a threshold for each sample between 100-150 [159].

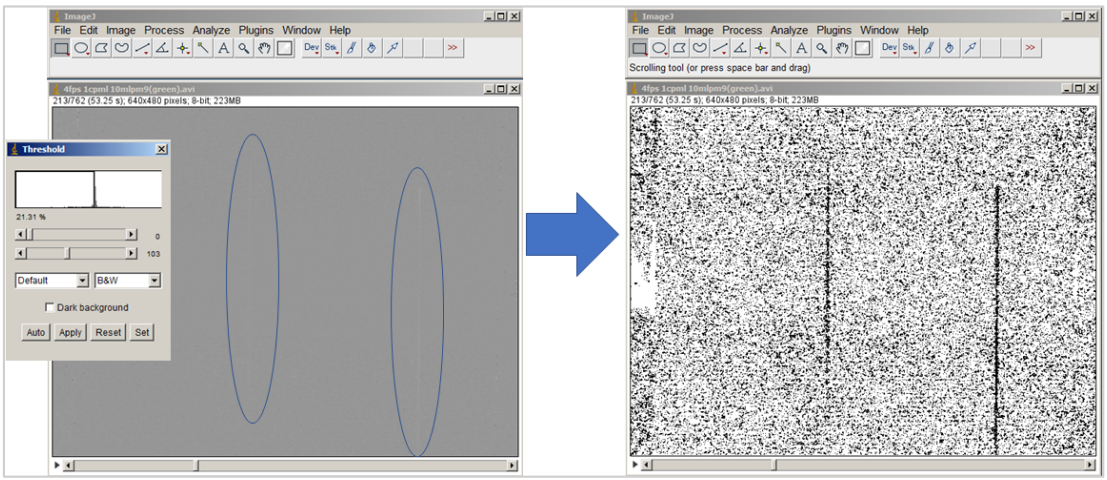


Figure 4.1: MTrack2 threshold selection*. The threshold was selected for each sample and it ranged between 100- 150. As noted in the figure, two streaks are clearly identified after thresholding the frame.*

To illustrate the importance of the threshold selection for MTrack2, figure 4.2 shows the difference between object detection using two different thresholds (for this example we created an artificial video file with four frames and two cells). MTrack2 returns a video file with centroid locations and labels from each object detected in each frame.

As show in figure 4.2a, MTrack2 could detect objects in both streaks in the frame (true positives, green ellipses), but in addition it picked up objects that don't correspond to streaks (false positive, red ellipses). With a more restrictive threshold (Figure 4.2b) MTrack2 did not pick up any false positive, but it did not detect the object for the streak in the left, producing a false negative result.

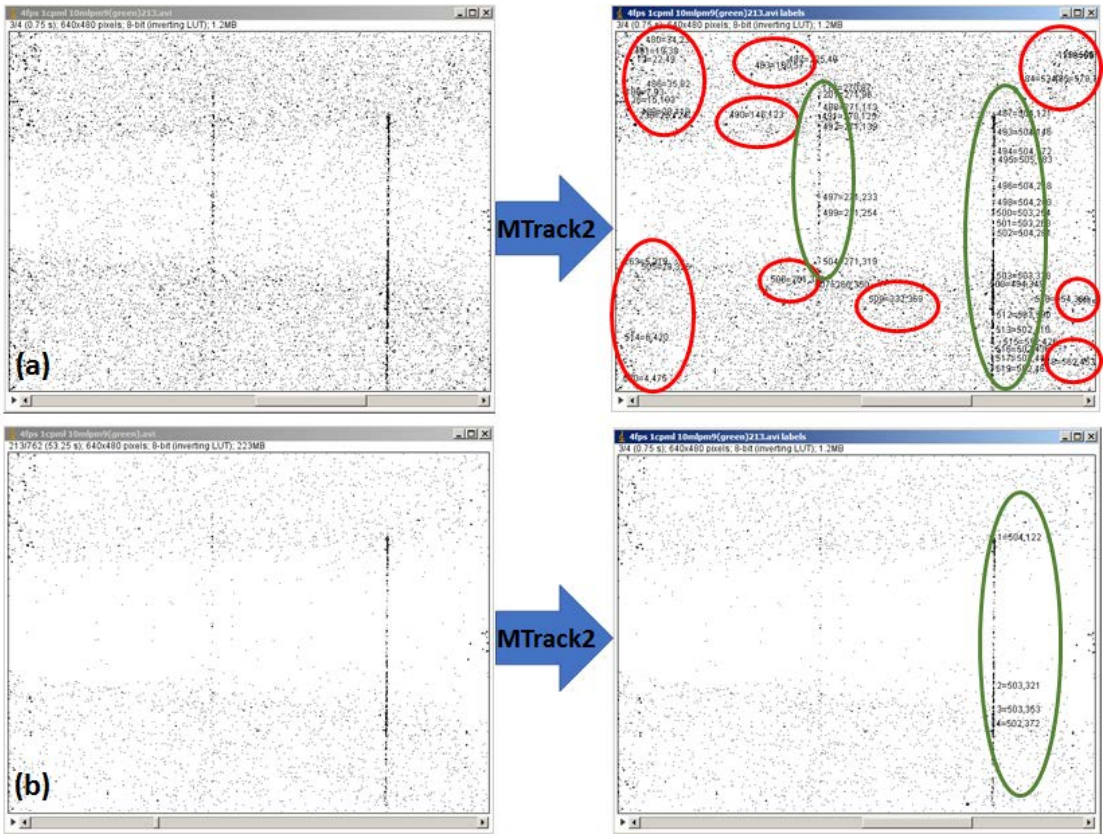


Figure 4.2: Comparing threshold selections*. Selecting a lower threshold enabled MTrack2 to detect both streaks but increases the false positive number (red circles) (a). Selecting a higher threshold decreases or eliminate false positives, but missed a true positive as shown in (b)*

MTrack2 provides an analysis output text file with the frame number, label and location of all objects detected. Figure 4.3(a) shows MTrack2 output from the video file with 4 frames to illustrate the process. Frame #2 shows 3 objects (Flags 1, 2 and 3). The same

objects were identified in Frame #3 plus an additional object (Flag 4). In addition, MTrack2 produces a summary (at the bottom of the form) including the track number, the number of the frames where the object was found and the distance in pixels traveled by the object. Figure 4.3(b) show the magnified area in the frames 2 and 3, where the objects were detected.

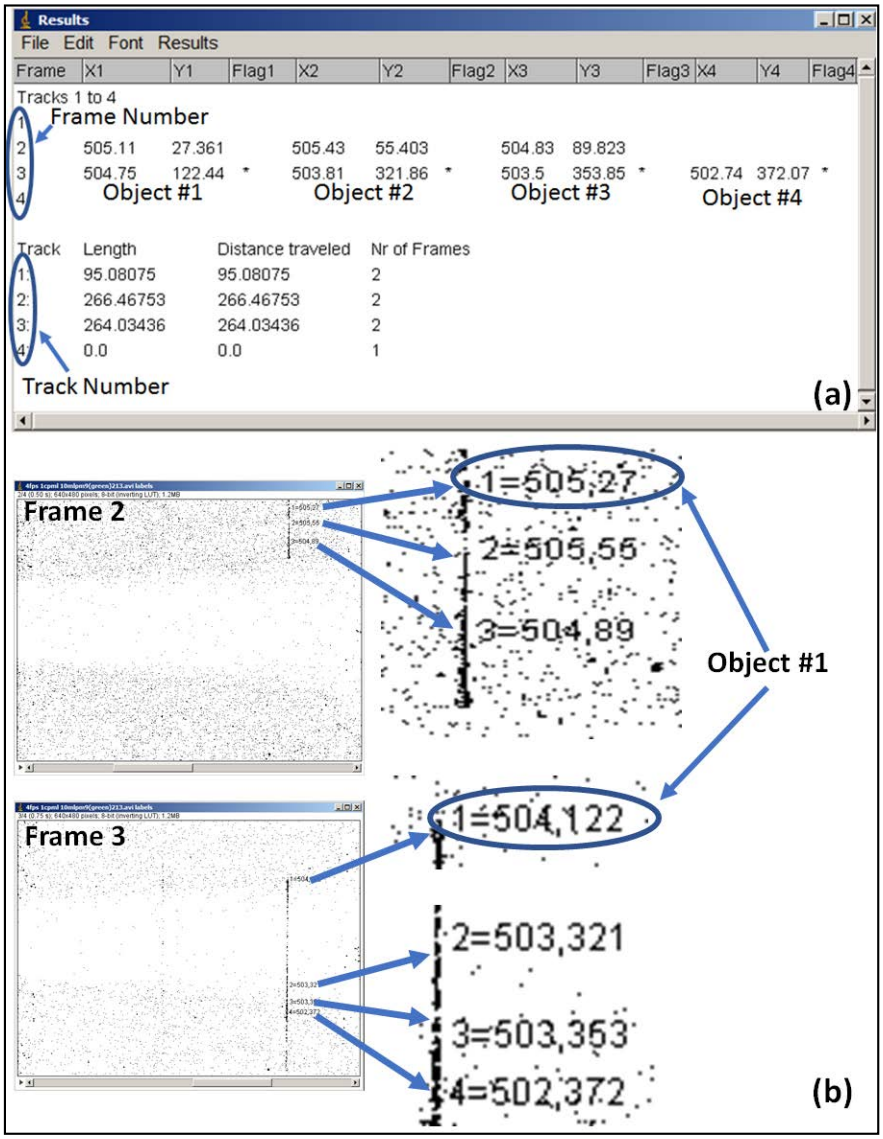
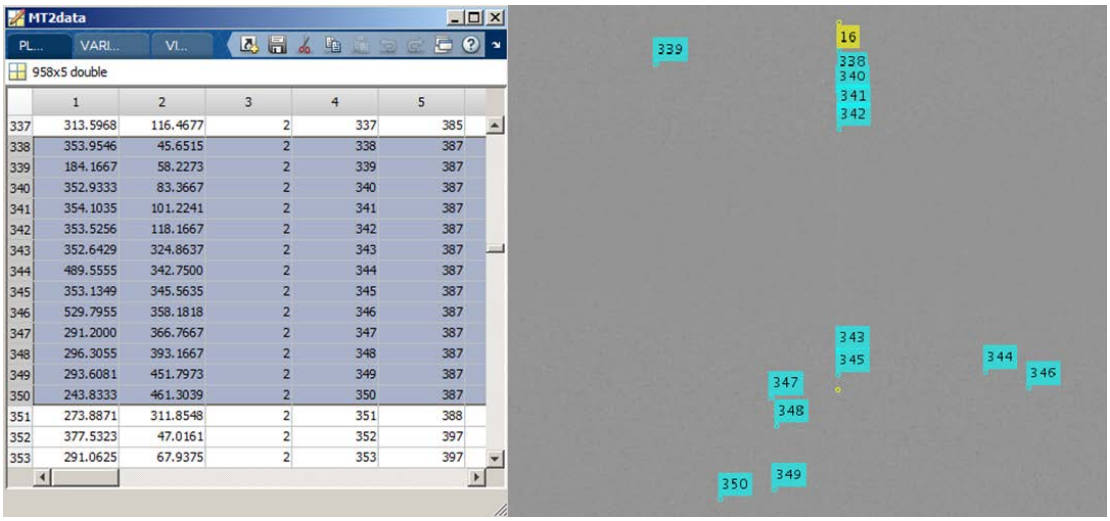


Figure 4.3: MTrack2 output*. (a) Text file produced by MTrack2 showing the frame number and the x,y pixel location of the centroid for each object in the frames. (b) Screen shot of the frames where the objects are located. In this case, all the objects were identified as belonging to the same cell.*

4.2.1 Counting cells

The output text file from MTrack2 is converted to MS Excel format and uploaded to MATLAB where is formatted and stored as a .mat file. Figure 4.4(a) shows the MATLAB formatting of the MTrack2 data for sample # 32: Columns 1 and 2 represent the "*x,y*" pixel locations of the objects centroids; column 4 represents the label and column 5 the frame number of the objects identified by MTrack2. In this example MTrack2 detected 13 objects in frame #387, (objects numbered from 338 to 350, column 4 figure 4.4(a)). Figure 4.4(b) shows the spatial distribution of the objects in the frame (frame 387).

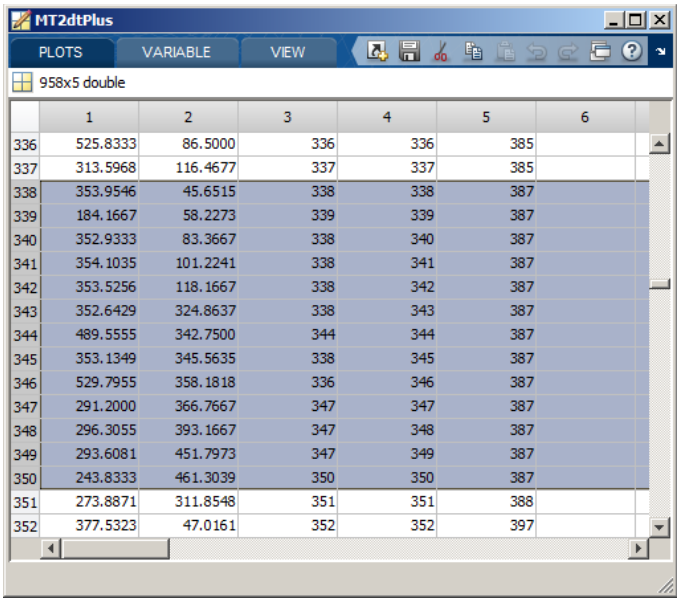


(a) (b)

Figure 4.4: MTrack2 output data and annotation. *(a) MTrack2 data formatted by MATLAB. Column 1 and 2 represent the centroid pixel (x, y) location respectively. Column 4 corresponds to the label assigned by MTrack2 and column 5 the frame number (column 3 is a place holder used in further processing). (b) Annotation using the object number provided by MTrack2 (column 4), over the ground truth. Object 338, 340, 341, 342, 343 and 345 matched with the ground truth cell #16.*

4.2.2 Association of the MTrack2 objects

The MTrack2 objects are grouped and relabeled according to the distance of their centroids: For objects in the same frame, all the objects within minimal centroid distance of 5 pixels in the *'x'* coordinate are labeled in the same group. Objects that appeared in one more of the next three consecutive frames showed minimal centroid distances of 7, 9 and 10 pixels respectively. The grouped label is stored in column #3 (Figure 4.5). Noted that MATLAB uses "Intrinsic Coordinates" as the default for the Image Processing Toolbox. The intrinsic coordinates *(x,y)* correspond to pixel indices, where the center point of a pixel is identical to the column and row index for that pixel. Therefore the upper left corner of an image is located at x=0.5, y=0.5 [160]. For example, objects 338, 340, 341, 342, 343 are grouped and relabeled as object 338 (column 3) and objects 347, 348 and 349 also are in the range of 5 pixels therefore are relabeled as object 347 (column 3).



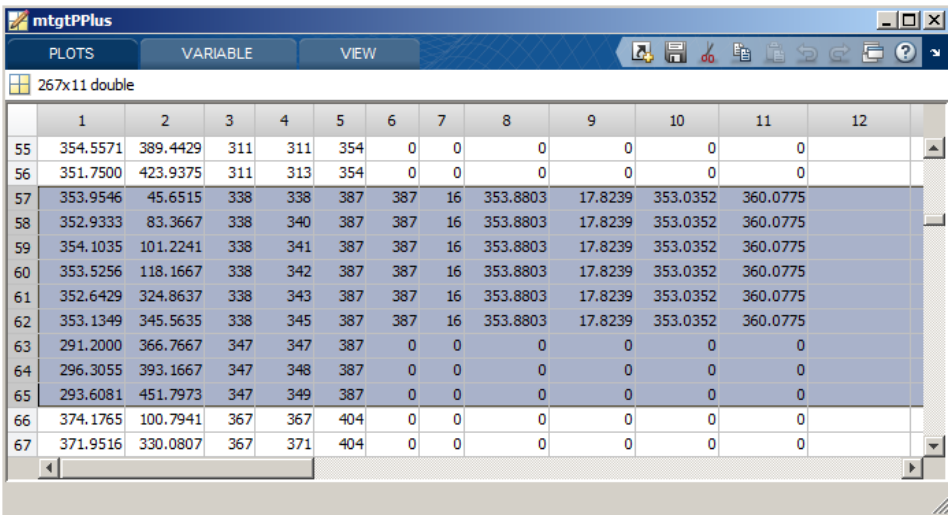
	1	2	3	4	5	6
336	525.8333	86.5000	336	336	385	
337	313.5968	116.4677	337	337	385	
338	353.9546	45.6515	338	338	387	
339	184.1667	58.2273	339	339	387	
340	352.9333	83.3667	338	340	387	
341	354.1035	101.2241	338	341	387	
342	353.5256	118.1667	338	342	387	
343	352.6429	324.8637	338	343	387	
344	489.5555	342.7500	344	344	387	
345	353.1349	345.5635	338	345	387	
346	529.7955	358.1818	336	346	387	
347	291.2000	366.7667	347	347	387	
348	296.3055	393.1667	347	348	387	
349	293.6081	451.7973	347	349	387	
350	243.8333	461.3039	350	350	387	
351	273.8871	311.8548	351	351	388	
352	377.5323	47.0161	352	352	397	

Figure 4.5: MTrack2 data. The objects are grouped and relabeled according to the 'x' coordinate of their centroid. Columns 1 and 2 represent x, y locations of the centroid; column 3, 4 and 5 represents the group label, the MTrack2 object ID and the frame number respectively.

4.3.3 Matching MTrack2 detected cells with the ground truth

All MTrack2 objects where the centroid is 5.0 pixels or less from the '*x*' location of the beginning or end of the ground truth streak (in the same frame) are considered to belong to the corresponding ground truth streak. For objects in the next three consecutive frames, the association centroid distance described above is applied, and the details of the matching procedure are described in the chapter "Sample Preparation and Ground Truth". In the case of objects not associated with ground truth streaks (false positives) we discarded all the objects that have only one appearance in the same frame. For example, in sample 32, frame 387 (Figure 4.5 above) objects 339, 344, 346 and 350 (column 4) were discarded.

Figure 4.6 shows the result of this procedure: Columns 6 -11 represent the ground truth information, as described in the chapter "Sample Preparation and Ground Truth", when an MTrack2 object match with the ground truth streak the MTrack2 objects is considered as true positive (TP) and the ground truth information is added to the corresponding row (columns 6-11). When no match is found, the object is considered a false positive (FP) and the default value (zero) is applied to the corresponding row. In addition, in the case of the FP objects, only objects with more than one appearance in the same frame are counted. For example, in column 3, only objects 338 (true positive) and 347 (False Positive) are considered for counting in frame 387, the rest of the objects were discarded.



	1	2	3	4	5	6	7	8	9	10	11	12
55	354.5571	389.4429	311	311	354	0	0	0	0	0	0	
56	351.7500	423.9375	311	313	354	0	0	0	0	0	0	
57	353.9546	45.6515	338	338	387	387	16	353.8803	17.8239	353.0352	360.0775	
58	352.9333	83.3667	338	340	387	387	16	353.8803	17.8239	353.0352	360.0775	
59	354.1035	101.2241	338	341	387	387	16	353.8803	17.8239	353.0352	360.0775	
60	353.5256	118.1667	338	342	387	387	16	353.8803	17.8239	353.0352	360.0775	
61	352.6429	324.8637	338	343	387	387	16	353.8803	17.8239	353.0352	360.0775	
62	353.1349	345.5635	338	345	387	387	16	353.8803	17.8239	353.0352	360.0775	
63	291.2000	366.7667	347	347	387	0	0	0	0	0	0	
64	296.3055	393.1667	347	348	387	0	0	0	0	0	0	
65	293.6081	451.7973	347	349	387	0	0	0	0	0	0	
66	374.1765	100.7941	367	367	404	0	0	0	0	0	0	
67	371.9516	330.0807	367	371	404	0	0	0	0	0	0	

Figure 4.6: Matching MTrack2 data with ground truth*. When a match is found (true positive) columns 6 to 11 are filled with the information from the ground truth. The default value when no match is found is zero. Columns 5 and 6 correspond to the frame number, column 7 corresponds to the cell number, and columns 8 to 11 represent the (x,y) location of the start and end of the streak respectively. Column 1and 2 represent the x,y location of the centroid; column 3 represents the group label and column 4 represents the MT2 object ID*

Figure 4.7 (sample 32 frame #387) shows the spatial distribution of MTrack2 objects compared with the ground truth. Objects 338, 340,341, 342, 343, 345 (Figure 4.7(a)) are relabeled as group object # 338 (Figure 4.7b) and determined to belong to ground truth cell # 16. Objects 348, 347 and 349 are relabeled as group object #347 (FP). Objects 339, 344, 346 and 350 were discarded.

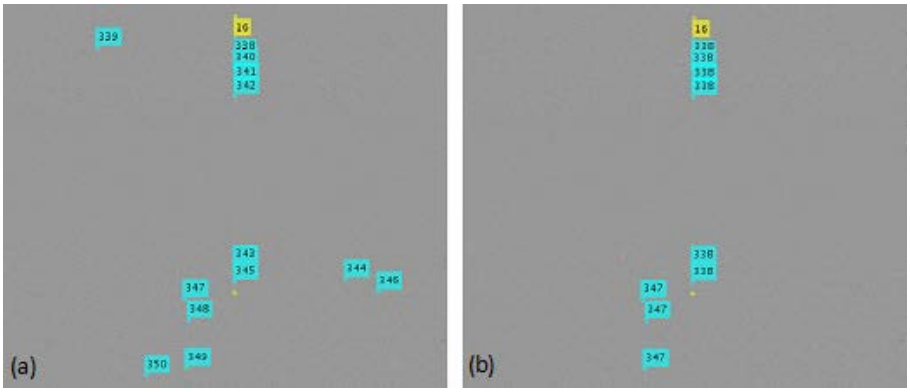


Figure 4.7: Distribution of MTrack2 objects*. (a) Spatial distribution of the MTrack2 objects. (b) The objects are relabeled and all objects with only one appearance in the frame are discarded*

4.3 CellTrack

CellTrack is an open source, self-contained, extensible and cross-platform software package developed at the Ohio State University in collaboration with the Middle East Technical University (Ankara, Turkey) [154]. CellTrack provides general purpose imaging processing and object segmentation methods and implements an edge-based method based on active contour models for detection and tracking of cell boundaries. CellTrack requires the input of the initial edge threshold and the linking threshold for efficient edge detection. For this dissertation, the initial edge detection threshold was set between 30-60, and the edge linking threshold was set between 20-50 for all samples as described in our previous work [159]. Figure 4.8 shows a screen shot of the CellTrack graphic user interface (GUI) showing two streaks crossing the frame (the same frame as shown in figure 4.1). Several objects from the streak on the right were detected. In the streak to the left only 2 objects were detected.

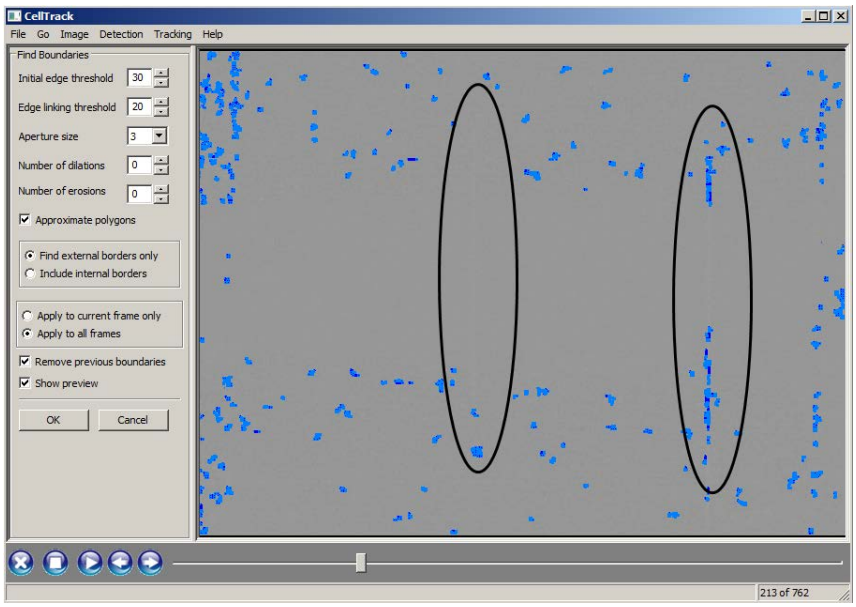
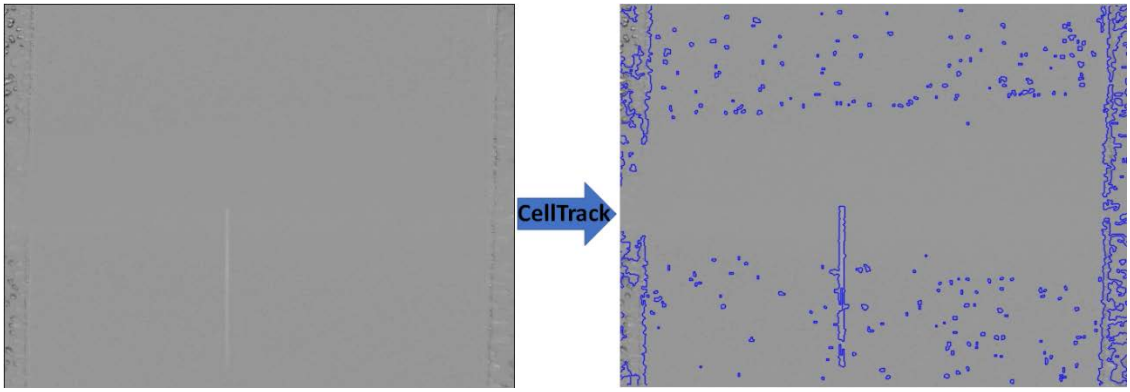


***Figure 4.8: CellTrack GUI** showing two streaks crossing the frame (black ovals).*

CellTrack provides an output video file (.avi) showing the objects detected in each frame (Figure 4.9). The samples were run through CellTrack and the .avi file containing the objects detected were saved for counting streaks.



a *b*

Figure 4.9: CellTrack object detection. (a) Original frame. (b) Frame after being processed by CellTrack

4.3.1 Counting cells

The .avi files containing the objects detected by CellTrack were processed using a MATLAB application developed for this purpose. All the objects within 40 pixels from the left and right boundaries of the frame were not included in the process. In addition, to eliminate small artifacts detected by CellTrack, all the objects with sizes less than 70 pixels were discarded. All the remaining objects are considered to be valid. A valid CellTrack object is defined as a streak that potentially belongs to a cell.

Using the MATLAB function *"regionprops()"* the centroid, bounding box, and size of the remaining objects were stored in an array (Figure 4.10). Figure 4.10 shows the properties of three objects detected in frame 731, where columns 1-2 represent the '*y*,

x' pixel coordinate of the centroid; columns 3-6 represent the bounding box; column 7 represents the size of the object; column 9 represents the frame number and column 8 is a sequential number assigned to each object in the frame.

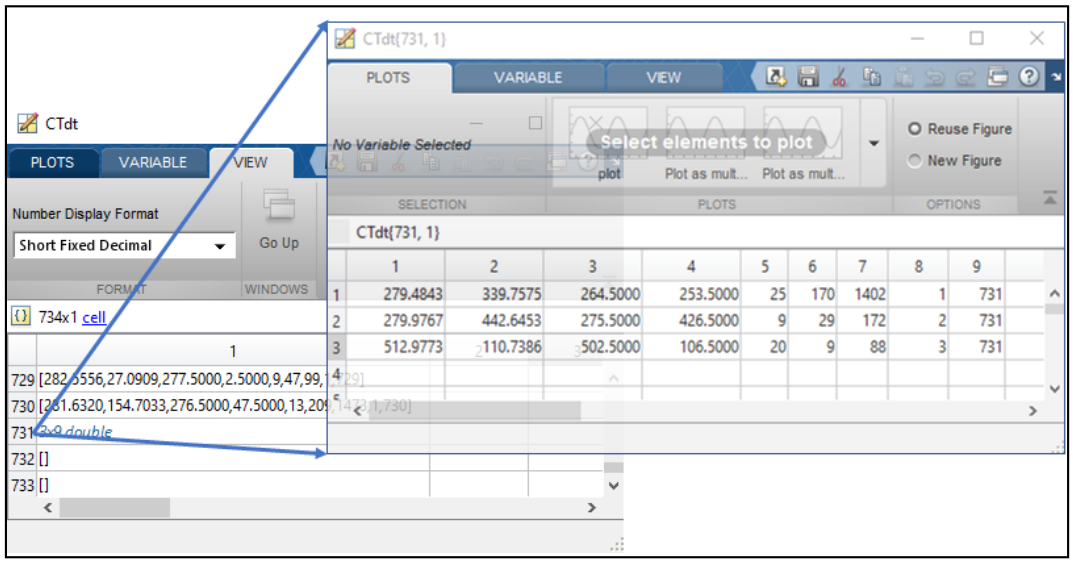
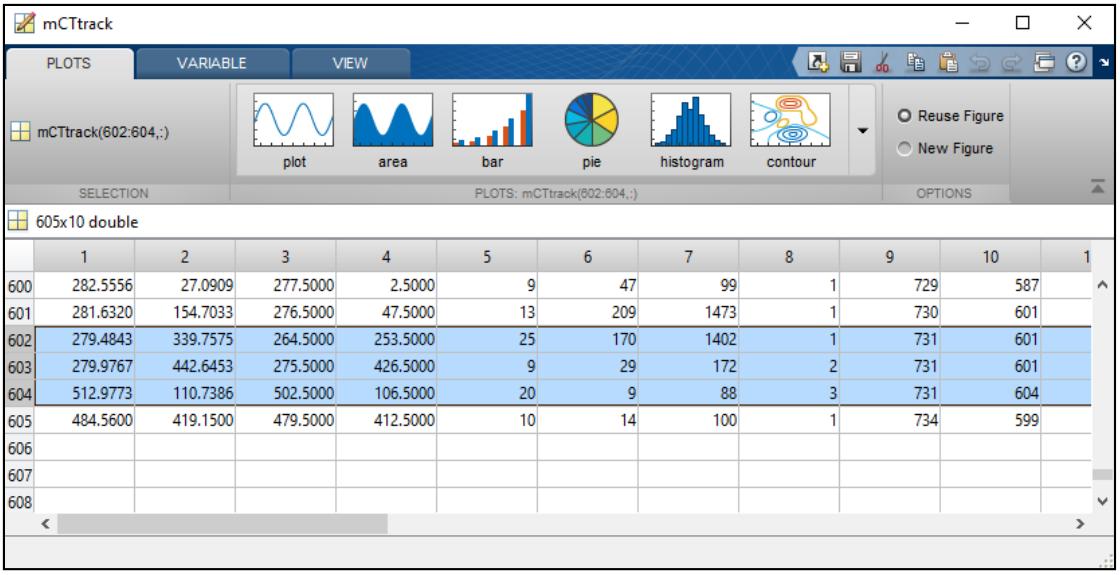


***Figure 4.10: CellTrack data**. Columns 1,2 represent 'x, y' location of the centroid; Columns 3-6 represent the bounding box; column 7 represents the object size; column 8 represents a sequential number of the objects in each frame and column 9 represents the frame number.*

4.3.2 Association of the CellTrack objects

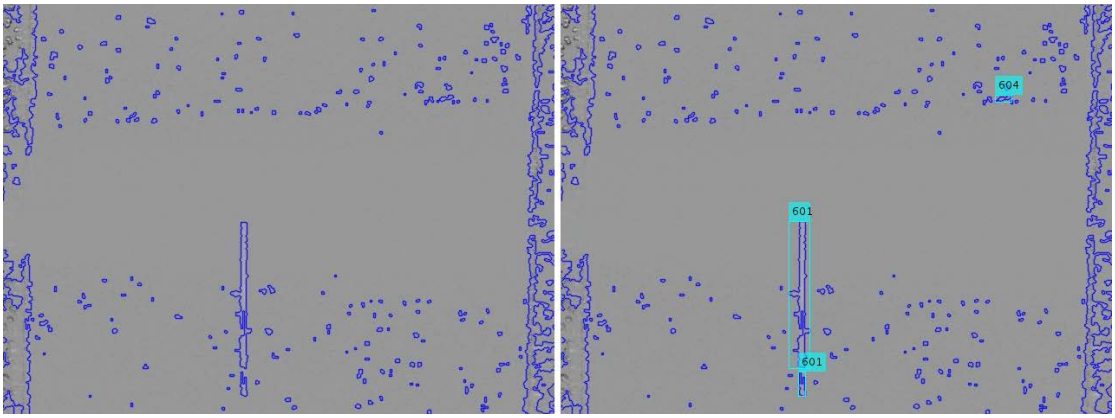
Using an approach similar to MTrack2, the objects were associated according the '*x,y*' pixel coordinates of their centroid (within minimal centroid association distance) to determine if they belong to the same streak. All the objects within the minimal association distance were grouped and relabeled. Figure 4.11 shows the result of this operation; Objects 1 and 2 in frame 731 were determined to belong to the same streak and were grouped and relabeled as streak 601 (Figure 4.11, column 10); object #3 was relabeled as streak 604.



	1	2	3	4	5	6	7	8	9	10	1
600	282.5556	27.0909	277.5000	2.5000	9	47	99	1	729	587	
601	281.6320	154.7033	276.5000	47.5000	13	209	1473	1	730	601	
602	279.4843	339.7575	264.5000	253.5000	25	170	1402	1	731	601	
603	279.9767	442.6453	275.5000	426.5000	9	29	172	2	731	601	
604	512.9773	110.7386	502.5000	106.5000	20	9	88	3	731	604	
605	484.5600	419.1500	479.5000	412.5000	10	14	100	1	734	599	
606											
607											
608											

Figure 4.11: CellTrack, objects association*. An additional column is added to indicate the group label (column 10)*

Figure 4.12 shows the spatial distribution of the objects detected in the frame as provided by CellTrack. Figure 4.12(a) shows all the objects detected by CellTrack. Figure 4.12(b) shows only the valid objects after eliminating the artifacts. Notice that in this example two objects were considered valid, but object 604 is probably a FP.



a *b*

Figure 4.12: CellTrack, spatial representation of the objects detected*. The figure shows that only three objects among all the objects detected by CellTrack in this frame are considered to belong to potential cells (valid objects). Two objects are considered to belong to the same streak and relabeled as streak #601*

4.3.3 Matching CellTrack detected cells with the ground truth

Using the ground truth (GT) and the centroid location, the CellTrack objects were associated with corresponding GT cells. Figure 4.13 below shows that CellTrack streak 601, corresponds to ground truth cell 21 (Figure 4.13 column 12). The zero in columns 11-16 indicate that these streaks cannot be associated to a true cell (false positive).

625x16 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
600	282.5556	27.0909	277.5000	2.5000	9	47	99	1	729	587	729	21	282.0493	2.6127	282.8944	49.0915
601	281.6320	154.7033	276.5000	47.5000	13	209	1473	1	730	601	730	21	281.2042	52.4718	280.3592	256.9789
602	279.4843	339.7575	264.5000	253.5000	25	170	1402	1	731	601	731	21	280.3592	253.5986	280.3592	456.4155
603	279.9767	442.6453	275.5000	426.5000	9	29	172	2	731	601	731	21	280.3592	253.5986	280.3592	456.4155
604	512.9773	110.7386	502.5000	106.5000	20	9	88	3	731	604	0	0	0	0	0	0
605	484.5600	419.1500	479.5000	412.5000	10	14	100	1	734	599	0	0	0	0	0	0

Figure 4.13: CellTrack, association of the streaks with the GT. *Columns 11, 12 represents the frame number and cell Id respectively. Columns 13-16 are the x,y location of the start/end of the GT streaks.*

In figure 4.14 (last step in the process), the frame with the identified streaks is overlaid with the corresponding ground truth frame. Figure 4.14 shows that streak number 601 matches with a cell in ground truth (true positive). On the other hand, object 604 cannot be associated with a ground truth streak and is deemed a false positive.

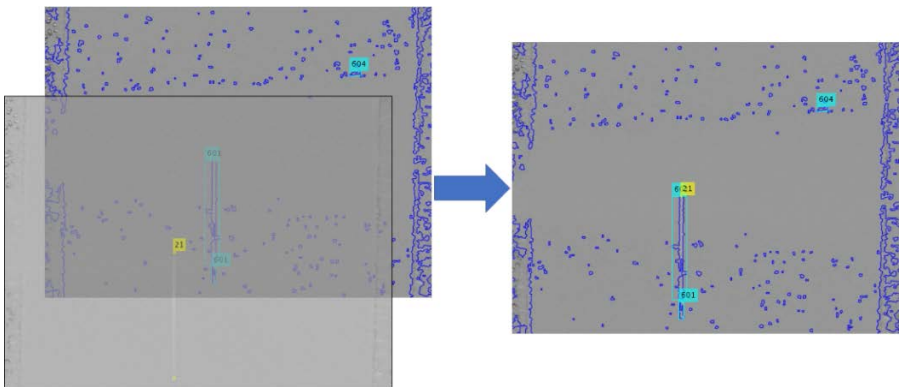


Figure 4.14. Association of a CellTrack object with the ground truth. *Object 601 was associated with cell 21(true positive). Object #604 is a false positive.*

Chapter 5: Sample Preparation and Ground Truth

5.1 Sample preparation

The webcam-based wide-field streak cytometer and the data used in this dissertation were reported in previous work [52,53]. Briefly, THP-1 monocytes were removed from a culture medium and the cells were labeled with SYTO-9 dye added to 1 mL of suspended cells. After labeling, cells were diluted to a level of approximately 1 cell/μL (measured by microscopy). From this solution, lower concentration samples of 1 cell/ml and 0.1 cell/ml were generated by single-step dilution. Samples were injected into the wide–field webcam–based streak mode flow cytometer (described in chapter II 5) in batch sizes of 30 ml with the flow rate set to 10 ml/min. The samples were imaged at 4 frames per second and the video files were saved as uncompressed AVI format.

5.2 Ground truth

In order to compare different cell tracking methodologies on the streak images, cells used for ground truth were visually identified for each sample using a MATLAB application developed to assist tagging and recording their locations while crossing the frames [159,161]. First, using the MATLAB *"VideoReader()"* function, the video files were converted to a *"VideoReader"* object and each frame was stored in a cell array where each cell contains a matrix representing a frame from the video file. Subsequently, the

cell array is loaded into a MATLAB application developed for this thesis *"Avi()"* where the frames can be visually inspected by a human for streaks cells.

The *"Avi()"* application facilitates tagging the streak and recording the frame number and the location of each streak by a click of the mouse (Figure 5.1).

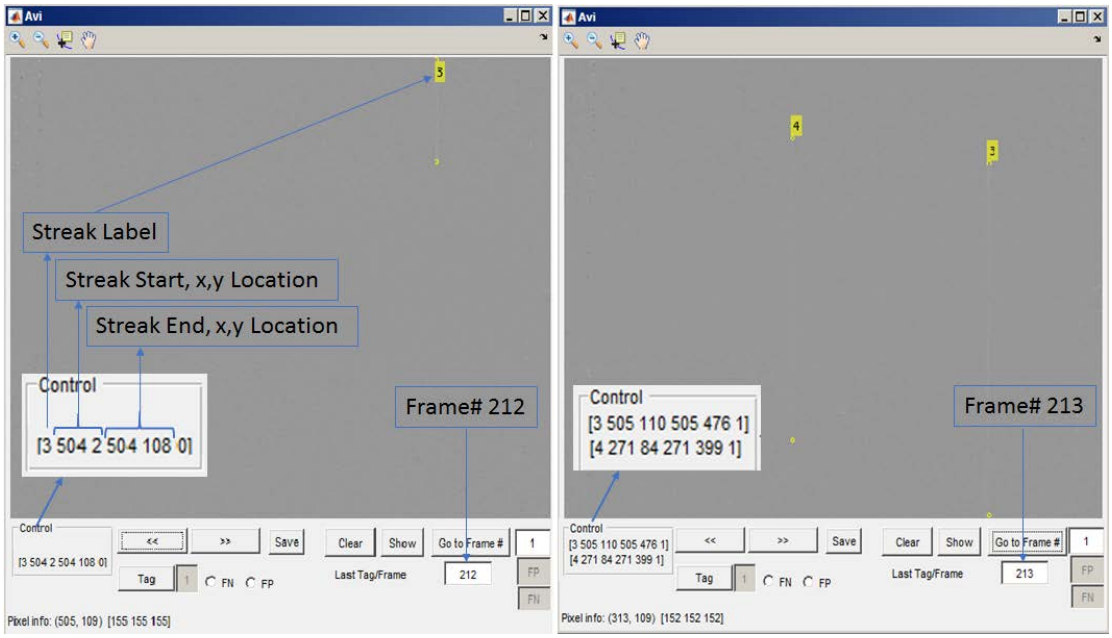


Figure 5.1: Avi() application GUI. *The figure shows 2 consecutive frames and the tagging of the streaks (by the click of the mouse). Avi() assists in the task of labeling and recording the location of visually identified streaks. The streak in frame 212 is the third streak in the sample (tag# 3) with x,y location starting in pixels 304,2 and ending in pixels 504, 108. The same streak (tag# 3) is also identified in frame 213 along with a new streak (tag# 4)*

The *Avi()* tool returns a video file with the ground truth streaks labeled, and a data file containing the tag number and the pixel location of the beginning and the end of each streak (Figure 5.2). This data file is used in subsequent steps to match the ground truth with different methodologies for cell detection and tracking.

After the cells are identified, labeled, and their location recorded using the *"Avi()"* tool, the next step is to determine the signal to noise ratio (SNR) of the ground truth cells.

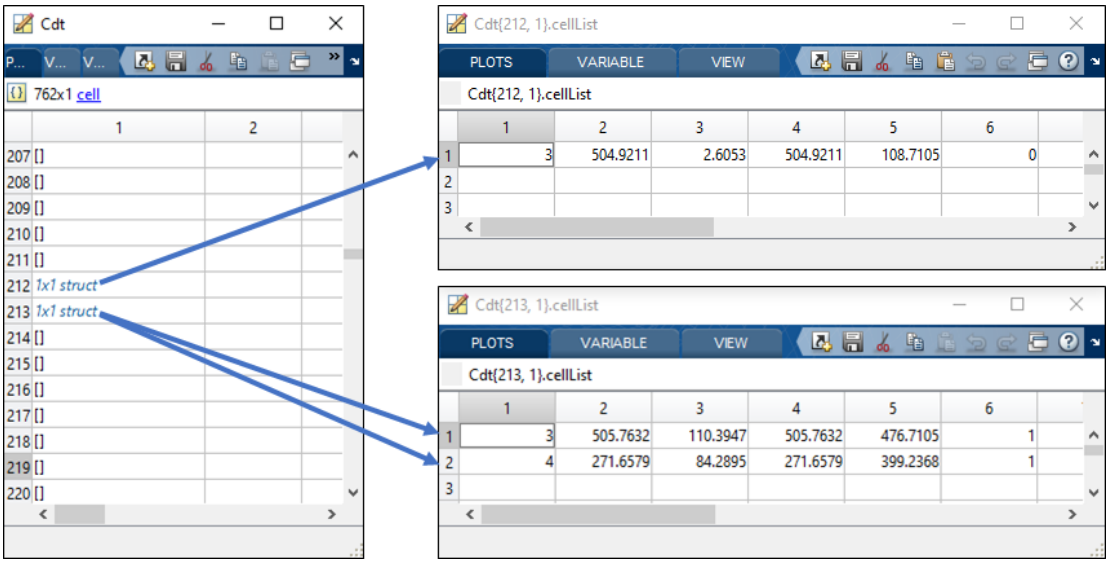


Figure 5.2: Avi() data file output*. Each row of the 'Cdt' file represents a frame in the sample and holds a structure containing the cell list with cell number and location for each streak in the list (Cdt.cellList). In this example row 212 on the 'Cdt' file (frame 212) holds a structure with one cell (cell #3), and row 213 holds the information for cell #3 and #4 in frame 213. In 'Cdt.cellList' Column 1 represents cell number/label, columns 2 and 3 represent 'x,y' pixel location of the end of the streak*

5.3 Signal to noise ratio (SNR) determination

The cells are enclosed in a bounding box and the SNR determined using the enclosed cells as signal. The SNR of a cell is defined as the maximum of the SNRs of all of its occurrences. The SNR of a cell's occurrence is computed from the intensity values of the pixels within its bounding box and a noise box. The noise box is defined as the area that consists of the 5 outer pixels of a 12 pixels dilation of the cell's bounding box (7 pixels away from the signal) (Figure 5.3). The signal consists of all the pixels enclosed within the cell's bounding box; this ensures that pixels associated with the cell are not used in estimating the background noise. The background noise is the average of the pixels in the rectangular frame (noise frame) surrounding the 5-pixel wide buffer rectangle and enclosed signal streak rectangle.

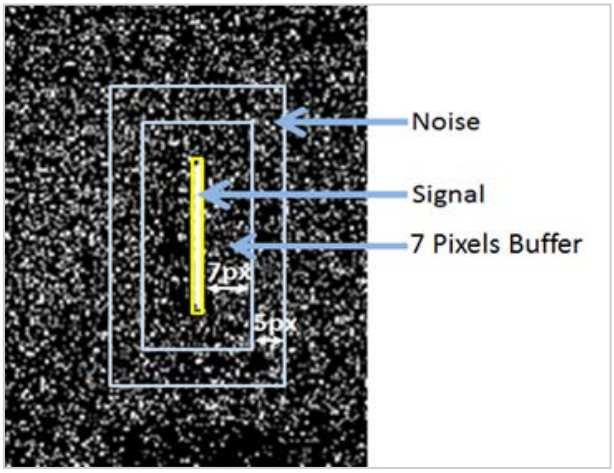


Figure 5.3: Schematic representation of the of the areas used to calculate SNR.

Using the indications above, the SNR is calculated as follow:

$$\boldsymbol{SNR = 10\ log_{10}\left(\frac{\sigma^2_{signal-bkg}}{\sigma^2_{noise}}\right)}$$

$$\textbf{where}\ \ \boldsymbol{bkg = \mu[noise]}$$

Using the formula above the SNR is calculated for each cell and the average SNR is calculated for each sample.

5.4 Matching detected cells to the ground truth

To facilitate the automatic evaluation of various cell detection/identification algorithms, we developed a MATLAB application for matching the detected cells to the ground truth cells (described previously [159,161]). First, a complete weighted bipartite graph is built with vertices consisting of all the detected cells *'D'* and all the ground–

truth cells *'T'* in all the video frames (Figure 5.4a), and with edges between *(D, T)* with weight equal to the minimum distance (across all the frames where both cells appear) between the bounding boxes (in the same frame) of the cells corresponding to *'D'* and *'T'* (Figure 4b). Next, we remove all edges with weight above a user provided threshold and replace the edge weights $w_{(D,T)}$ with $w_{(max)} - w_{(d,t)}$, where $w_{(\max)}$ is the maximum edge weight (Figure 5.4c). Finally, we compute a maximum weight matching in the resulting graph (Figure 5.4).

This matching procedure associates each detected cell with at most one ground truth cell. Note that some detected cells may not match any ground truth cell; in which case, we consider such detected cells to be "false positives" *(FP)*. Similarly, some ground truth cells may not be matched with any detected cell; in which case, we consider such ground truth cells to be "false negatives" *(FN)*. The matched detected cells are considered "true positives" *(TP)*.

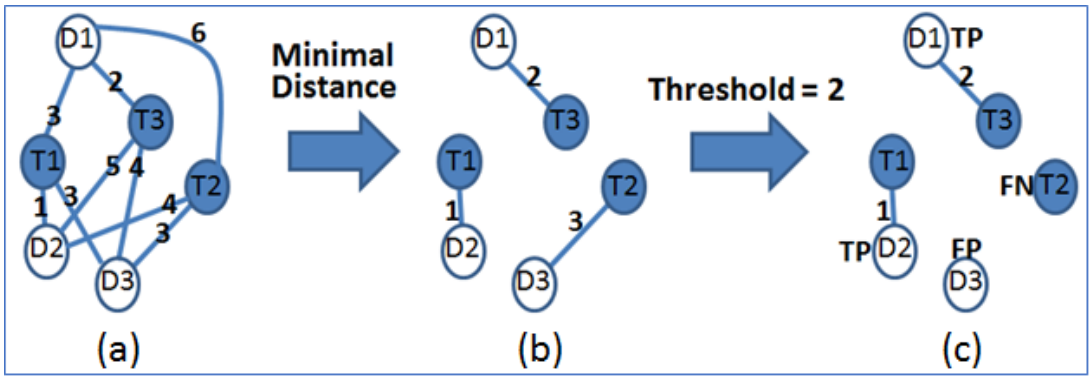


Figure 5.4: Matching detected cell with the ground truth.. *(a) Complete weighted bipartite graph using the distance as weight. (b) Minimal weigh is used to match Detected and True cells. (c) Using a user defined threshold, all the weights above the threshold (in this case distance of 2) are removed and the matched cells are classified as true positive (TP), the unmatched true cells are classified as false negative (FN) and the unmatched detected cells are classified as false positive (FP)*

Upon counting the *FN*, *FP*, and *TP* cells, several performance metrics are computed: the *precision*, *false negative rate (FNR)*, sensitivity (or *recall*), and the F1 score as shown below. These performance metrics are used to evaluate the cell detection/identification algorithms developed in this work (Table 5.1). Note that true negatives are not practical nor relevant in our context (since any sequence of boxes of pixels that does not match a ground-truth cell is a true negative).

Table 5.1: Performance metrics used in this dissertation

Performance metric	**Definition**
Number of True Positives	TP
Number of False Positives	FP
Number of False Negatives	FN
Precision (or Positive Predictive Value)	$Precision = \frac{TP}{FP + TP}$
Recall (or Sensitivity)	$Sensitivity = \frac{TP}{FN + TP}$
Miss Rate (False Negative Rate)	$FNR = \frac{FN}{FN + TP}$
False Discovery Rate	$FDR = \frac{FP}{FP + TP} = 1 - Precision$

Chapter 6: Automated Streak Detection Algorithm

6.1 Streak detection algorithm

The streak detection and tracking algorithm in this chapter was reported in our previous work [161]. The algorithm uses a decision rule based on geometrical and intensity distribution (GID) features e.g. intensity, length, SNR etc. to classify cells as spurious and non-spurious. The center of mass (centroid) of the bounding box of the streaks is used to track the cells in consecutive frames. The process identifies candidate cells from streaks and matches detected cells to ground truth cells. The algorithm is implemented in MATLAB R2014b and consists of three major procedures: (1) binary mask creation for all potential streak locations, (2) candidate cell identification based on the binary mask developed in the previous step, and (3) true cell identification by filtering out spurious candidate cells. These steps are illustrated in the flow chart in figure 6.1. After identification as true cells, they can be matched to the GT cells.

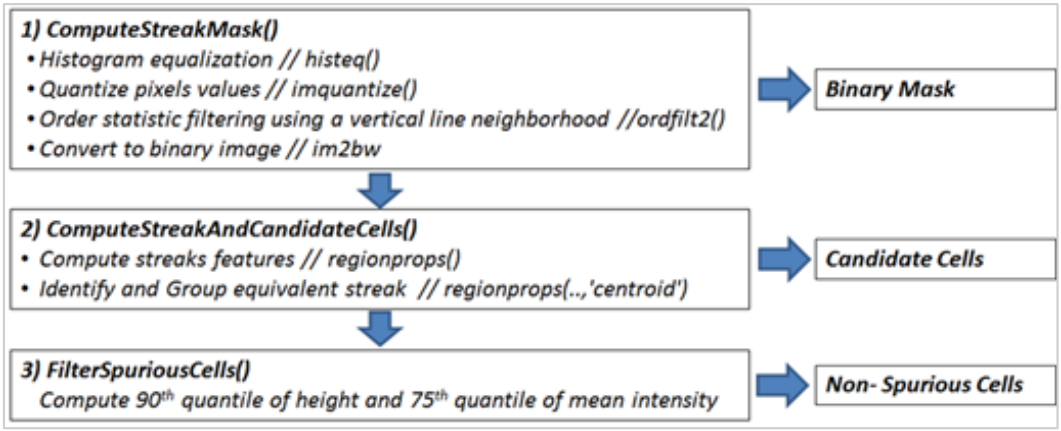


Figure 6.1: Flow Chart: Streak detection and cell counting Algorithm. *Three-step process for automated streak detection and cell counting: (1) finding streaks, (2) identifying candidate cells, and (3) filtering out spurious cells to identify true cells from the streaks.*

6.1.1 Streak detection and binary mask; Procedure I, ComputeStreakMask()

The objective of this procedure is to create a binary mask with the location of all the streaks in representing cells (procedure I, "ComputeStreakMask."). The major steps of this procedure (in MATLAB-like pseudocode) are given below. Streaks are defined as vertical elements that are expected to belong to cells. In this step we identify all potential streaks through thresholding using the Otsu method[132] and noise reduction using a 2D order-statistical filter.

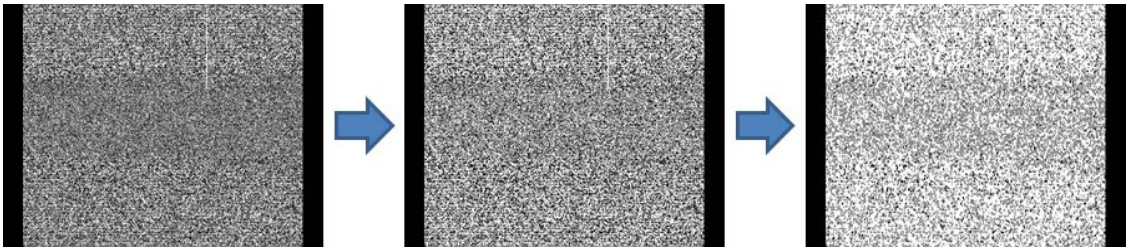
Procedure I: ComputeStreaksMask
Input: Gray image frame I and parameters prm
Output: Binary image mask B

1: $\mathrm{I}(:, prm.cutOff) \longleftarrow 0$ // Zero left and right margins of frame
2: $\mathrm{I} \longleftarrow$ histeq(I) // Histogram Equalization
3: $\mathrm{I} \longleftarrow$ imquantize(I, mutlithresh(I, 2)) // quantize pixel values
4: $\mathrm{I} \longleftarrow$ imclose(I, strel('rectangle', [3 2])) // close with short rectangle
5: $\mathrm{I} \longleftarrow$ ordfilt2(I, prm.K, getnhood(strel('line', prm.lineLength, 90))) // order-statistics filtering using a vertical line neighborhood
6: $\mathrm{B} \longleftarrow$ im2bw(I) // convert to binary image
7: $\mathrm{B} \longleftarrow$ bwareaopen(B, $prm.areaSize$) // remove small connected components
8: $\mathrm{B} \longleftarrow$ imopen(B, strel('line', prm.maskLineLength, 90) // Remove short components
9: **return** B

Step 1: The 640x480 pixel frames are preprocessed by removing 40 pixels in left and right margins to eliminate artifacts shown in the margins of the frame (Figure 6.2a).

Steps 2-3: The image intensity is adjusted by histogram equalization, then two threshold values (of the image intensity) are selected using Otsu's method to quantize the image into three levels (Figure 6.2b).

Step 4: To fill small gaps along the boundary of the foregrounds objects, a morphological "close" with a 3x2 pixels rectangle is slid across each frame. This generates unwanted noise (Figure 6.2c), but better defines the foreground objects (no gaps along the boundary).

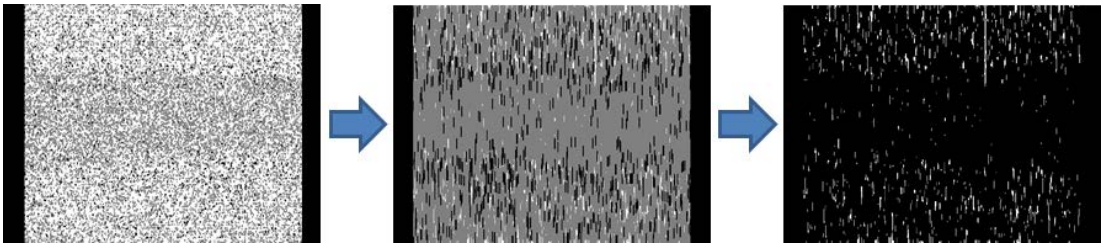


(a) ***(b)*** ***(c)***

Figure 6.2: Streak detection algorithm, Procedure I, steps 1-4*. Notice in (c), the streaks are hidden by high noise generated by the procedure to fill small gaps along the boundary of foreground objects.*

Step 5: In order to reduce the background noise generated in the previous steps, a 2D order statistical filtering is performed. Briefly, a 23x1 pixels rectangle is slid across the frame, replacing the pixel value at the rectangle origins with the 3rd smallest of the pixel values contained in the rectangle. The result of this operation will eliminate pixels with low values around the foreground object but will preserve pixels with small values inside the foreground object (Figure 6.3a, 6.3b).

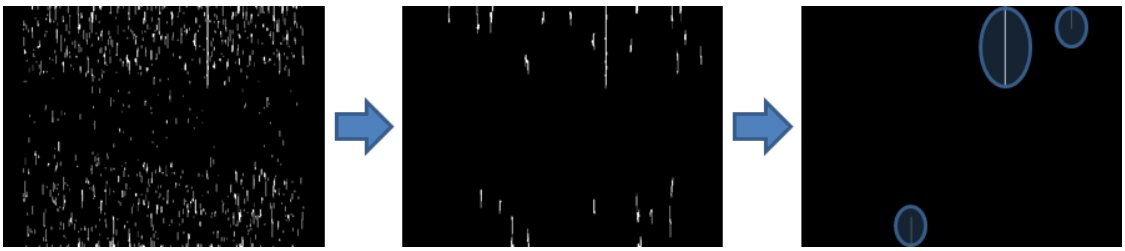
Step 6: The three-level threshold image is converted to a binary image (Figure 6.3c).



(a) *(b)* *(c)*

Figure 6.3: Streak detection algorithm, Procedure I steps 5, 6*. (a) resulting high noise image from the previous step, (b) 2D order filtering was used to reduce background noise and (c) the three- level image is converted to a binary image.*

Steps 7, 8: These steps are necessary to clean small artifacts from the image and eliminate extrusions from the boundary of foreground objects. All objects (potential streaks) smaller than 50 pixels are eliminated (Figure 6.4a, b), then a morphological open, with an 81x1 pixels rectangle, is performed to remove non-vertical short objects, which most probably represent artifacts. The final result of this procedure is a binary mask with the location of all the streaks that potentially represent cells (Figure 6.4c).



(a) (b) (c)

Figure 6.4: Streak detection algorithm Procedure I, steps 7,8*. All the foreground objects smaller than 50 pixels along with non-vertical short objects are discarded. In addition, all the extrusions of the boundary of foreground objects are eliminated providing a clean, well defined mask of detected streaks (c). Notice the binary mask (c) includes three streaks, but only the streak in the middle corresponds to a real cell.*

6.1.2 Identifying candidate cells; Procedure II, ComputStrkAndCandteCells()

The goal of this step is to identify candidate cells using the streak location in the binary mask. This is achieved by procedure II, "ComputeStreakAndCandidateCells."

Procedure II: ComputeStreaksAndCandidateCells
Input: Video of gray frame $\mathbf{I}$ and parameters prm
Output: Set of candidate cells in the video frames.

for each frame $\mathbf{I}$ in the video **do**
  $\mathbf{B} \longleftarrow$ ComputeStreaksMask($\mathbf{I}, prm$) // construct binary mask to focus the search for streaks
  for each connected component S (i.e. streak) in $\mathbf{B}$ **do**
    compute statistics (features) of $\mathbf{B}$ restricted to S
    compute statistics (features) of $\mathbf{I}$ restricted to S
  end for
end for
identify all pairs of streaks that are 'equivalent' (i.e. the displacement of the centroid of their bounding boxes is within specified tolerances)
partition the streaks into equivalence classes based on the above equivalence relation
identify each block of streaks in the above partition as a candidate cell
for each candidate cell c **do**
  compute cell features by aggregating features of the streaks associated with that cell
end for
return set of candidate cells together with their associated streaks and their computed features

The binary mask from the previous step is overlaid on the original image and each streak representing a potential cell is enclosed in a boundary box for further possessing (Figure 6.5). The following features are computed for each streak: (a) area, (b) bounding box (centroid, height, and width), (c) major and minor axes length of enclosing ellipsoid, (d) eccentricity, (e) orientation, (f) perimeter, and (g) descriptive statistics (min, max, media, quantiles, variance, sum) of the values of the streak's pixels. Most of these features are provided by the "regionprops" MATLAB command. Streaks (across all frames) are grouped and labeled as equivalent if they are expected to belong to one cell

based on the displacement of their centroids within a tolerance level. Using their centroid and orientation, streaks are partitioned into equivalent classes and identified as a candidate cell (equivalent streaks are defined as streaks that belong to the same cell). The streaks that are in the same equivalence class are now identified as candidate cells. For each candidate cell, we compute aggregates (min, max, mean, median, range, variance) of the features of its streaks. Note that in MATLAB's image coordinate system, the x–axis (y–axis) runs along the image's width (height), increasing from left–to–right (top–to-bottom) with the (0, 0) point at the upper-and-left-most pixel. By the end of this step the streaks are annotated and the candidate cells are identified for each frame.

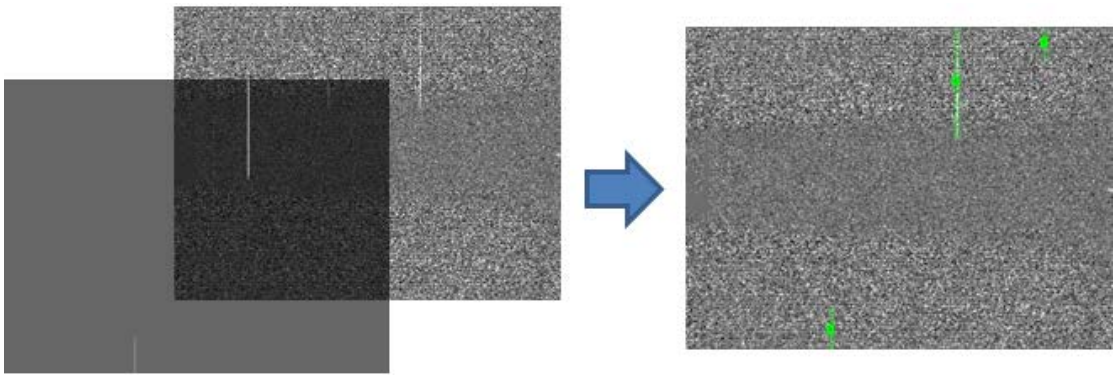


Figure 6.5: Overlaying the binary mask with the original image*. Based on the binary mask, three objects are identified and classified as candidate cells (green lines) in the original frame.*

6.1.3 Identifying true cells; Procedure III, FilteringSpuriousCells()

The candidate cells identified in the previous steps are evaluated according to intensity and length. The goal of this procedure is to eliminate cells with low probability of being true cells. The major steps of this procedure are listed below:

Procedure III: FilterSpuriousCells

Input: Set of candidate cells together with the streaks in each frame of a video file

Output: Set of non–spurious cells.

1: find subset J of cells with the following features

($max\_MeanIntensity > 160$ and $max\_MajorAxisLength >= 125$) or

($max\_MeanIntensity > 200$ and $min\ Centroid.Y > 150$ and $max\_StreakLength < 350$)

2: retain in J only the cells with

($max\_MeanIntensity > 160$ and $max\_MinorAxisLength < 10$ and $max\_MajorAxisLength > 110$)

3: **for** each video frame with at least 10 streaks **do**

4: compute the 90th quantile of the height and the 75th quantile of the MeanIntensity of the streaks in the frame

5: remove from J those cells whose streak in the frame has height or mean intensity below these quantiles

6: **end for**

7: **return** set J of cells

At this point a decision rule is applied to eliminate spurious cells and identify the candidate cells with a high probability to be real cells. Cells are evaluated according to their intensity, length, width and displacement of the centroid in the '*Y*' direction.

Step 1: Create a subset of cells (subset '*J*') that have mean intensity greater than 160 and length greater or equal to 125, or mean intensity greater than 200 and centroid displacement in '*Y*' direction greater than 150 and length less than 350.

Step 2: Retain in this subset '*J*' only cells with intensity greater than 160; and width less than 10; and length greater than 110.

Step 3-5: In these steps the decision rule includes the number of cells per frame. Due to the nature of the rare cells (since it is expected to be a small number of cells per frame) if more than 10 cells are found in a frame, only the cells with length greater than the 90th quantile and intensity greater than 75th quantile are retained in '*J.*' Conversely, in frames with fewer than 10 cells, all the cells are retained in '*J.*' A graphic representation of the decision rule is show in figure 6.6.

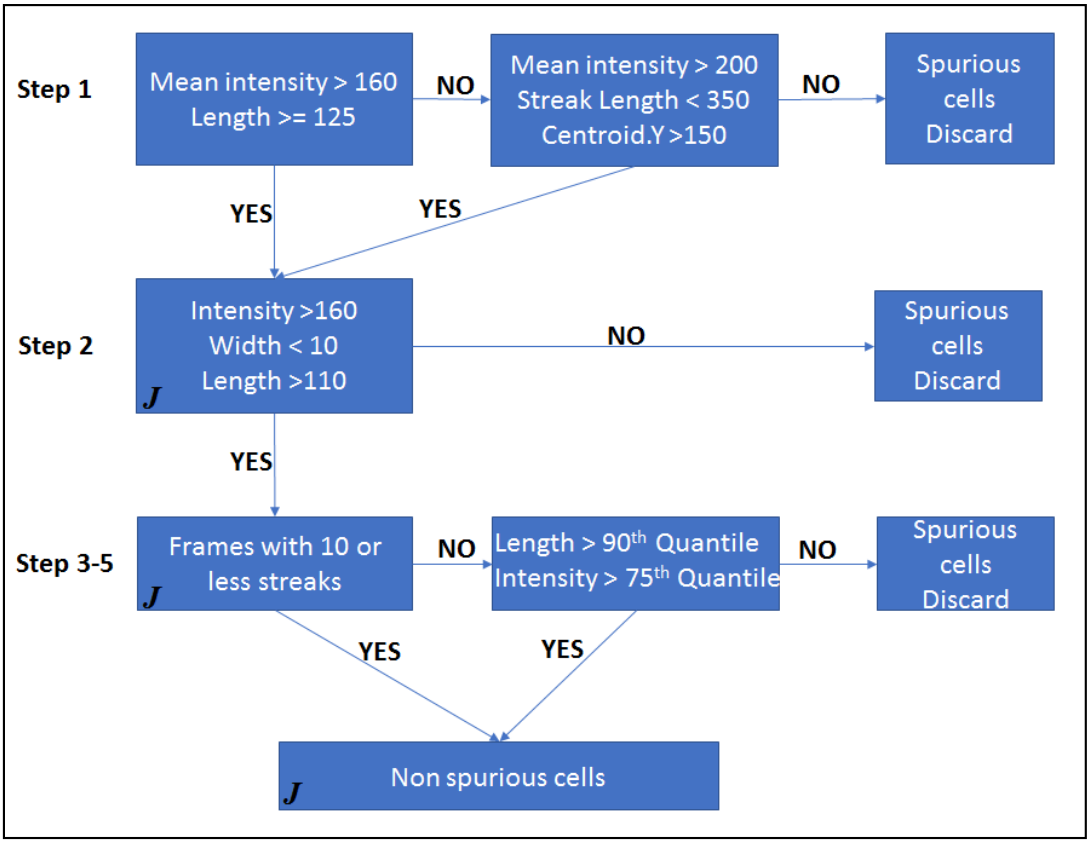
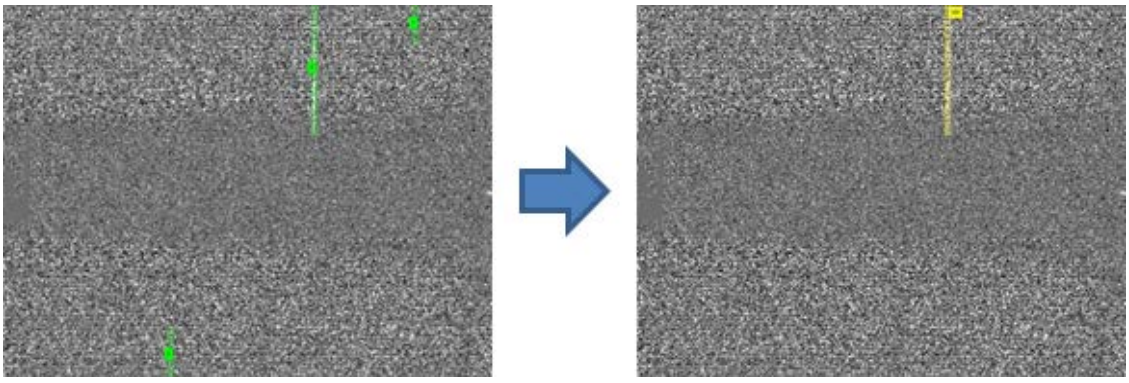


Figure 6.6: Decision rule, procedure III. *The purpose of the desion rulre is to eliminate Spurious cells.*

The results from procedure 3 are shown in figure 6.7. As the result of the previous procedure, three candidate cells are identified, but only one candidate cell is selected as non-spurious according to the decision rule in procedure 3.



(a) (b)

Figure 6.7: Eliminating Spurious cells. *Only one object was classified as a non-spurious cell(b) from the three objects identified in the previous steps (a).*

6.2 Performance results of the streak detection algorithm

The streak algorithm was evaluated in video files from 27 samples of 30 ml with a nominal concentration of 1 cell/ml. The SNR was determined for each ground truth cell and the average SRN for each sample was calculated. Each detected cell was compared with the ground truth and classified as TP, FP or FN as described in the "Sample Preparation and Ground Truth" chapter. Using the TP, FP and FN count, the FPR, TPR, FNR and Sensitivity was calculated for each sample. Table 6.1 shows the results of the streak algorithm compared with the ground truth. The samples are ordered by the average SNR of the ground truth cells.[161]

Table 6.1: Samples ordered by SNR of the ground truth cells (1cell/ml): ID= Sample ID; FP = False Positive; FN=False Negative; TP = True Positive; FPR = False Positive Rate; FNR = False Negative Rate; TP = True Positive Rate; GT= Ground Truth; SNR Signal-to-Noise Ratio.

ID	FP	FN	TP	GT	FDR	FNR	Sensitivity	Precision	F1 score	SNR
27	2	8	12	20	0.14	0.40	0.60	0.86	0.71	2.86
28	1	8	22	30	0.04	0.27	0.73	0.96	0.83	3.03
35	3	17	13	30	0.19	0.57	0.43	0.81	0.57	3.41
34	6	22	7	29	0.46	0.76	0.24	0.54	0.33	3.42
32	0	10	13	23	0.00	0.43	0.57	1.00	0.72	3.50
31	3	10	19	29	0.14	0.34	0.66	0.86	0.75	3.55
29	1	14	16	30	0.06	0.47	0.53	0.94	0.68	3.71
23	4	6	28	34	0.13	0.18	0.82	0.88	0.85	4.05
26	2	11	26	37	0.07	0.30	0.70	0.93	0.80	4.08
30	1	14	11	25	0.08	0.56	0.44	0.92	0.59	4.15
33	0	8	17	25	0.00	0.32	0.68	1.00	0.81	4.27
24	5	2	27	29	0.16	0.07	0.93	0.84	0.89	4.41
17	4	4	20	24	0.17	0.17	0.83	0.83	0.83	4.52
25	1	4	29	33	0.03	0.12	0.88	0.97	0.92	4.56
22	1	3	24	27	0.04	0.11	0.89	0.96	0.92	4.58
15	1	1	22	23	0.04	0.04	0.96	0.96	0.96	4.64
20	1	5	26	31	0.04	0.16	0.84	0.96	0.90	5.02
21	3	3	18	21	0.14	0.14	0.86	0.86	0.86	5.08
19	2	4	25	29	0.07	0.14	0.86	0.93	0.89	5.09
18	1	3	28	31	0.03	0.10	0.90	0.97	0.93	5.37
13	1	3	27	30	0.04	0.10	0.90	0.96	0.93	5.91
12	4	0	21	21	0.16	0.00	1.00	0.84	0.91	6.02
16	0	1	20	21	0.00	0.05	0.95	1.00	0.98	6.20
14	0	1	30	31	0.00	0.03	0.97	1.00	0.98	6.25
11	2	1	25	26	0.07	0.04	0.96	0.93	0.94	6.46
9	1	3	24	27	0.04	0.11	0.89	0.96	0.92	7.42
10	2	1	24	25	0.08	0.04	0.96	0.92	0.94	7.44
Average					**0.09**	**0.22**	**0.78**	**0.91**	**0.83**	

Table 6.1 shows that the average *precision* for the streak algorithm is 91%, however the average *sensitivity* of the algorithm is only 77%, matched by the high *False Negative Rate* (22%).

Ordering the samples by SNR reveals that the sensitivity of the algorithm is affected by the SNR of the sample. Figure 6.8 shows the distribution of the samples according to SNR and *sensitivity*; the red dots represent samples with SNR grater or equal to 4.41 dB (higher SNR), the green dots represent samples with SNR lower than 4.41 dB (lower

SNR). The two clusters can be easily distinguished indicating a potential cut off value for the *sensitivity* of the algorithm based in the SNR of the samples.

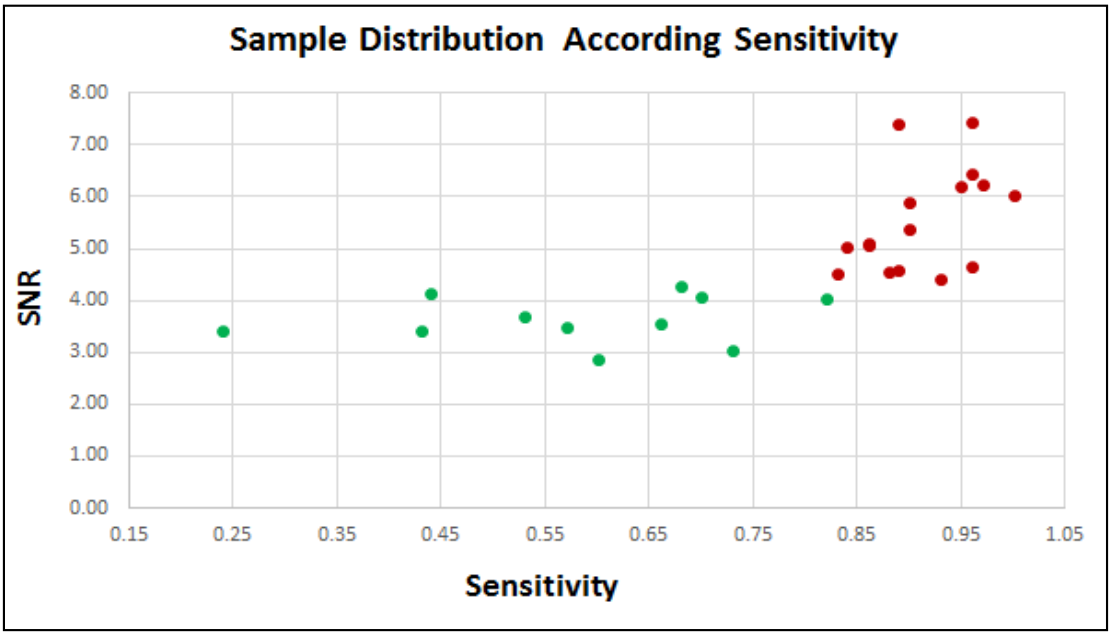


Figure 6.8: Sample distribution according to Sensitivity vs SNR. *The samples can be clustered in two groups: The red samples have SNR grater or equal to 4.41dB and the green samples have SRN lower than 4.41dB.*

Table 6.2 shows average values of FNR, precision, sensitivity and F1 score grouped by average SNR. For samples with an SNR equal or greater than 4.41 dB, the algorithm has sensitivity greater or equal to 83% (Table 6.1) with an average precision of 91% (Table 6.2), and for samples with SNR lower than 4.41 dB the average Sensitivity is 58%. This finding indicates that samples with SNR equal or greater than 4.41 dB would be desirable for the best performance of the algorithm. [161].

Table 6.2: Samples classified into two groups according to their SNR (1cell/ml). The figure shows average false negative rate (FNR), precision, sensitivity, and F1 score.

SNR	Average *FNR*	Average *Precision*	Average *Sensitivity*	Average *F1 score*
< 4.41 dB	0.42	0.88	0.58	0.69
>=4.41 dB	0.09	0.93	0.91	0.92
All SNRs	0.22	0.91	0.78	0.83

Our algorithm was further evaluated by comparison with two current cell tracking tools MTrack2 and CellTrack (Table 6.3). The streak detection algorithm performed better than MTrack2 and CellTrack for the detection of cells in the 1 cell per ml samples. Sensitivity, precision and F1 score for all SNR samples were considerable better in the streak detection algorithm than the other two methods. [159].

Table 6.3: Comparison between the streak detection algorithm (1cell/ml) with two methods for cell tracking.

Method	SNR< 4.41 dB			SNR >= 4.41 dB			All SNR		
	Sensitivity	Precision	F1 score	Sensitivity	Precision	F1 score	Sensitivity	Precision	F1 score
CellTrack	0.26	0.19	0.22	0.60	0.29	0.38	0.46	0.25	0.31
MTrack2	0.43	0.27	0.30	0.71	0.80	0.74	0.59	0.58	0.56
Streak Algorithm	0.58	0.88	0.69	0.91	0.93	0.92	0.78	0.91	0.83

One of the issues of the streak detection algorithm is the need for the user to pre-define values for the decision rule such as: intensity over 150, length over 110, width less than 10, etc. Predefined values are selected for a specific cohort and need to be modified if variations of the image acquisition procedures are introduced (e.g., changes in flow rate or exposure time) because they will affect the size and speed of the streaks. A more adaptive procedure would be desirable to avoid this issue. For example, in Step 4, an

evaluation of all the streaks in the frame is done to calculate the optimal intensity at the 90^{th} quantile and optimal streak length of the 75^{th} quantile to determine if the cells should be kept or discarded. These estimations are done without consideration of the user's predefined values and taking into consideration all the cells in the frame, which can result in a better way to discriminate between spurious and non-spurious cells. More nearly optimal cut off values for intensity or length could yield more nearly accurate identification of cells increasing true positive detections and decrease false negative detections, as well as decrease the number of false positive detections. Finally, the algorithm performed better with higher values of Signal-to-Noise Ratio (SNR), which underscores an unmet need a better cell classification system for low SNR samples. Analysis of the performance of the algorithm in samples with low cell concentrations (samples with 1 cell per 10 ml) in the presence of greater amounts of debris that produce high (overwhelming) numbers of spurious cells in comparison with the extremely low concentration of non-spurious cells raises another issue.

Our results motivated a need to develop a more adaptive algorithm with less dependence of high SNR that can accurately identify and discriminate true cells from artifacts. This motivation led us to develop a "Relational Streak Detection Algorithm" discussed in the next chapter. The relational streak algorithm improves upon the streak algorithm in this chapter by integrating the following three modifications to the algorithm.

a) Enhance the decision rule implemented in the previous version of the algorithm by use of geometrical and intensity distribution (GID) features to create an image classifier through that uses several machine learning routines.

b) Incorporate visual word features extracted by the bag of features (BoF) method to create an image classifier based on a visual vocabulary.

c) Integrate relational features extracted from the images to create a selective permeable filter. This is a concept analogous to selective cell membrane permeability, in which only certain molecules are allowed to enter or exit a cell. The computational equivalent of this concept is described in the next section.

Chapter 7: Relational Streak Detection Algorithm

The relational streak algorithm combines the geometrical and intensity distribution (GID) features (used in the previous version of the algorithm) with visual words features extracted by the bag of features (BoF) and relational features.

7.1 Geometrical and intensity distribution (GID) and visual words features

These features were extracted by the previous version of the algorithm and include streak height, width, orientation, streak bounding box area, pixel intensity, signal to noise ratio among others. Using a machine learning routine, an image classifier is created based in GID features. The machine learning methods were selected by the available method at the MATLAB classification learner App.

Visual word features were extracted from the streak images using the MATLAB bag-of-feature method to create a visual dictionary. All the candidate cells were used to create the image dataset for this purpose.

7.2 Image dataset

The image files were created cutting a rectangle using 10 pixels from the left and right of the centroid, and the length of the bounding box of the streak. This creates a 21 pixels width rectangle with the length corresponding to the length of the streak. Figure 7.1 shows a ground truth streak and the corresponding jpg file and label. The jpg files were

labeled using the sample, number, frame number and cell ID. These image fragments of the candidate cells were stored in two different folders corresponding to two image categories: Ground truth images *'Te'* for true positive candidate cells and non-ground truth images *'Fe'* (false positive candidate cells). Using this procedure, 1077 *'Te'* images and 24602 *'Fe'* images from the 1 cell per ml samples were created.

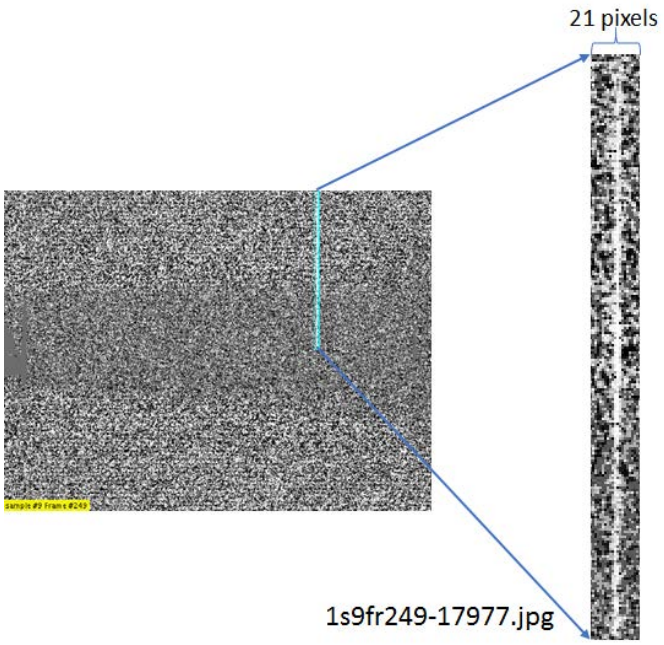


Figure 7.1: Streaks images for BoF. *The image database is created from the streaks identified previously by the binary mask and classified as candidate cells by the decision rule on the previous version of the streak algorithm. Using the centroid location, a 21 pixels wide square containing the streaks is stored as jpg file and labeled according to sample number, frame number and cell ID. The ground truth and non-ground truth images are stored in different folders for training the image classifier. Notice that in this example the actual bounding box of the streak is 7 pixels wide.*

7.3 Bag of features (BoF)

Using the MATLAB function *"bagOfFeatures()"*, the Speeded-Up Robust Features (SURF) from the two images categories (*Te, Fe*) were extracted. The feature point

location for the two categories are selected using the Grid method, and the features are extracted from the point location using a GridStep of [8 8] and block widths of [32, 64, 96, 128].

After the stronger features are extracted from all the images, a 500 words visual dictionary is created using K-Means clustering.

For our samples, around 50,000 SURF features were grouped in 500 clusters where each cluster represent a word in the image dictionary. Figure 7.2 illustrates this process using the MATLAB function *"detectSURFFeatures()"* for extracting the SURF features and the creation of the 500 word visual dictionary represented in the visual words histogram.

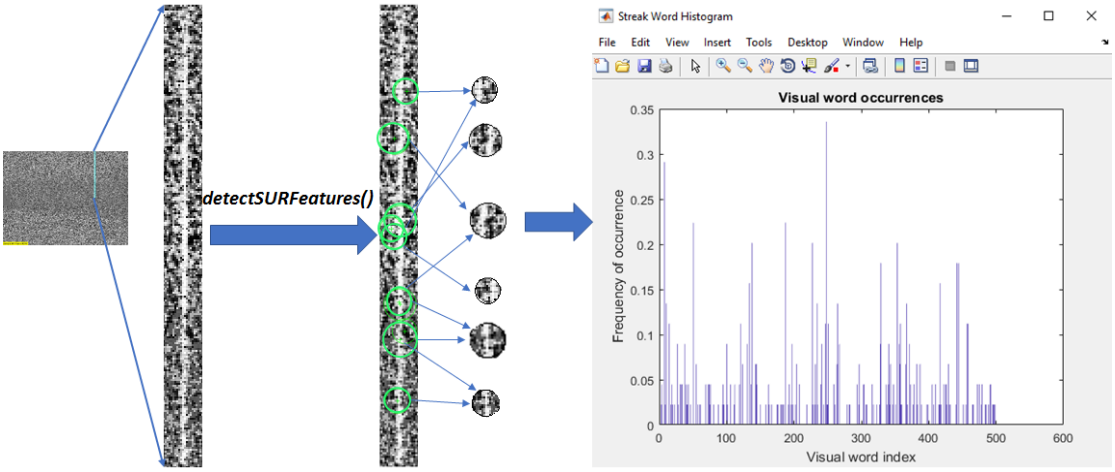


Figure 7.2: Visual words histogram. *To illustrate this procedure, the Features were extracted using the MATLAB function "detectSURFFeatures()" and clustered in 500 word vocabulary represented in the visual word dictionary. In the figure, the 10 stronger features detected by the SURF algorithm are represented by green circles.*

7.4 Relational features

The classifiers used until this point only consider individual streaks to eliminate spurious candidate cells. These classifiers do not take in consideration the relation between the streaks that represent a cell, or the spatial location of the streaks from different cells. For example, the vertical edges of wide objects can be mistaken as two cells running in parallel (similar 'y, x' coordinate location and similar height); since the chance that two cells appear in the same frame is low, the chance that two cells running in parallel is even lower, therefore parallel candidate cells have a great chance of being spurious cells. However, this effect can be mimic by a true cell running beside an artifact. Figure 7.3 shows two frames with parallel streaks, the frame in the left show several debris structures, but only one was picked up as a 2 candidates cells.

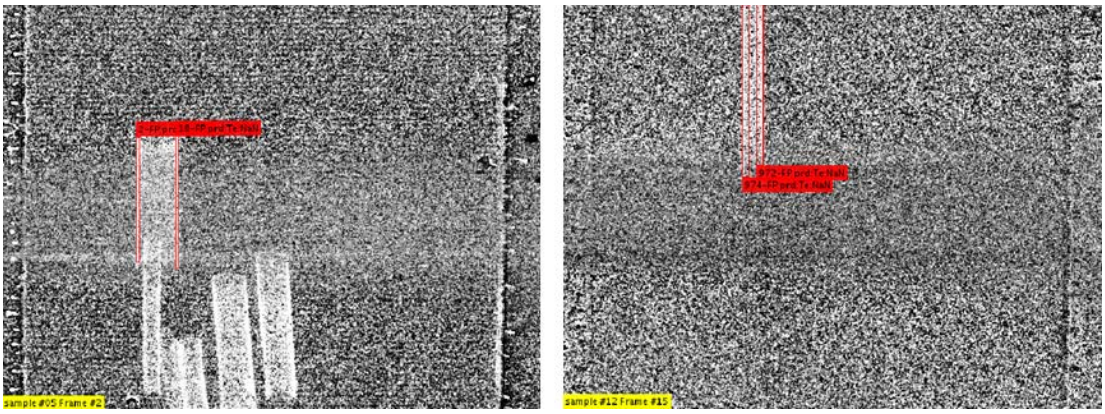


Figure 7.3: Parallel streaks*. The edges of wide objects can produce apparent parallel streaks. Since the streaks in this figure have similar location and similar height, they are classified as parallel streaks and considered spurious.*

Another example is the width ratio of the streaks of a cell. Streaks in a non-spurious candidate cell can have variable lengths, but their width should be similar, therefore

candidate cells with significance variation on the width of their streaks are suspicious to be spurious. Figure 7.4 shows a candidate cell with two streaks with considerable different widths, these candidate cells have a high probability of being spurious cells as is confirmed by visual inspection.

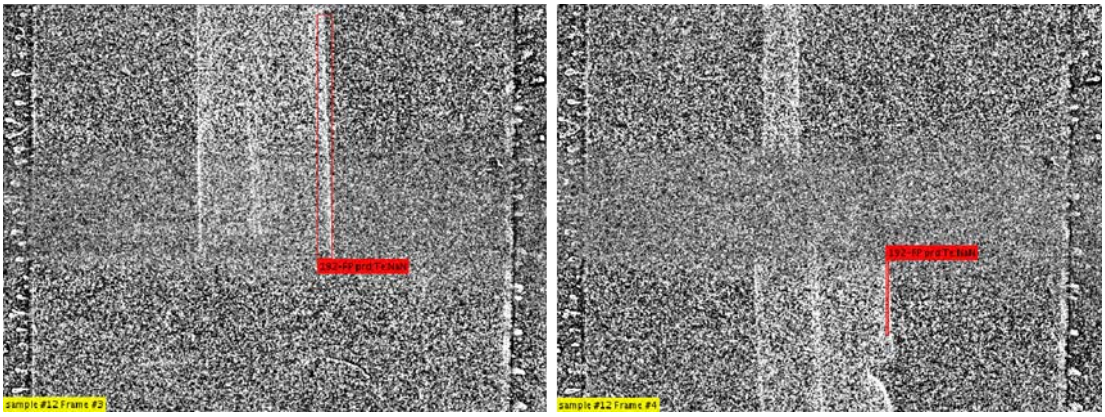
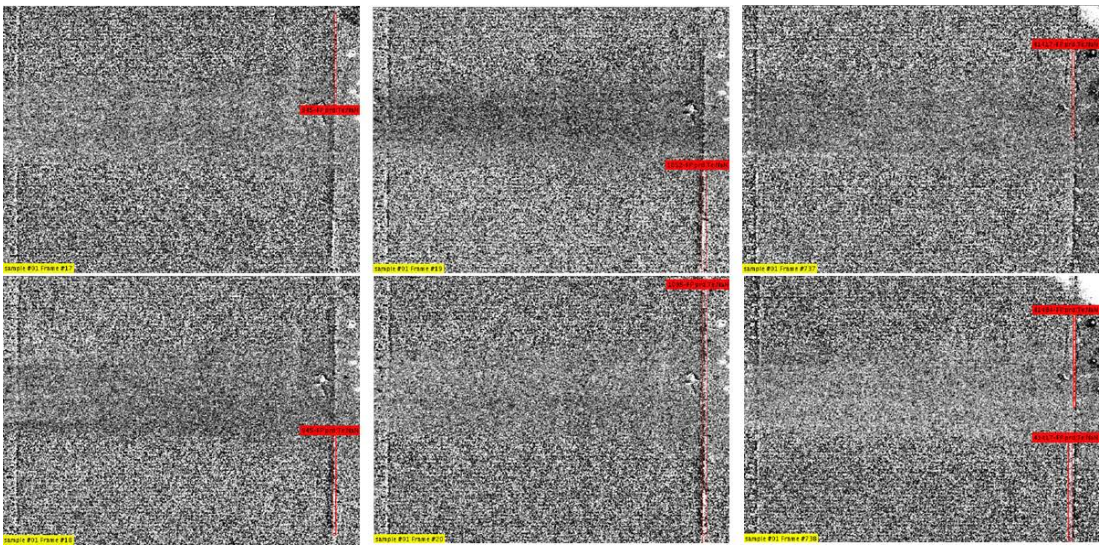


Figure 7.4: Streaks with considerable different width. *Candidate cell number 192 in frames 3 and 4 have 2 streaks with considerable different width to be classified as a non-spurious cell*

In addition, some artifacts tend to appear in similar location in different frames, this is especially noted in the edges of the frame due to the way that the image is obtained (Figure 7.5). To identify these artifacts, the location of the streaks in different frames needs to be taken in consideration. For the identification of these artifacts, the edges of the frame were partition in 8 vertical segments of 8 pixels wide e.g. [8, 16, 24, 32, 40, 48, 56, and 64] along all the frames of sample. If many candidate cells appear in the same vertical partition, it might indicate the presence of an artifact in that section of the frame across the sample.



a *b* *c*

Figure 7.5: Stationary cells*. Six different frames showing steaks in a partition location. a) Frames 17 and 18 shows two streaks in 'x' pixel location 597 and 594, these streaks can be easily mistaken as non-spurious, but frames in (b) 19 and 20 and (c) 737 and 738 also have streaks in the same partition ('x' pixel location: 594,594, 593,594 respectively) indicating that they are spurious cells*

Another relational characteristic of the candidate cells that can be used as a relational feature is the number of candidate cells in a frame. Due to rare nature of cells, is expected to find only one candidate cell for every few frames, for example, as mentioned in chapter I, blood from a metastatic cancer patient can contain as little as 1 circulating tumor cell per ml. Each sample is 30 ml flowing at 10 ml per minute through the device's flow chamber with the camera capturing 4 frames per second; consequently, cells can appear in no more than 3 frames, therefore it is expected to find no more than 3 consecutive streaks (1 cell) for every 33 frames. If a frame contains only one streak, the frame can be considered "*clean*" and the candidate cell in the frame has a good chance to be a non-spurious cell. For the classification purpose, it is a good

idea to protect candidate cells in *clean* frames from the intermediate filtering layers and defer the classification decision to the highest layer.

Finally, there is another important characteristic of the cells images that can be used as a relational feature, the number of streaks that represent a candidate cell. Due to the length of the streaks, non-spurious candidate cells are expected to appear in 2 or, in some cases, 3 consecutive frames (each occurrence in the frame is represented as one streak), therefore most of the non-spurious candidate cells are represented by more than one streak, but there are many single streak candidate cells or SSCs (candidate cell represented by only one streak). For example, around 3 out of 4 non-spurious candidate cells are non-SSCs (in the 1 cell per 10 ml samples), in contrast in the same data set this proportion is reversed and 3 out of 75 spurious cells are non-SCCs. Spurious SSCs represent more than 96% of all spurious candidate cells and more than 92% of all candidate cells in the sample, therefore SSCs are highly suspicious to be spurious. However, caution must be taken using this feature to avoid discard non-spurious SSCs. One advantage of non-SSCs is that they can afford to lose one of its occurrences and still be correctly classified by its other occurrences; since SSCs do not have this advantage, SSCs should be protected from intermediate filtering layers to avoid the chance to eliminate non-spurious SSCs by mistake. This advantage is showed in the prediction protocol in cases of classification conflicts e.g. when the final classifier ended up classifying one occurrence of a candidate cell as *'Te' (*true cell) and another occurrence from the same candidate cell as *'Fe'* (in this case is the function of the classification protocol to eliminate the conflict). Since only few candidate cells have

more than 2 occurrences, ties are very common and in the case of ties, the classification protocol favors positive selection and solve the conflict declaring all the occurrences for that candidate cell as *'Te'*. One issue with relational features is that they cannot be computed by standard classifier based on geometrical or intensity distribution features, or by visual words (bag of features) of individual cells alone and in isolation from the other cells; therefore, a separate filter will be necessary to incorporate these features into the algorithm. As will be discussed in the next section, the bag of feature method yields excellent sensitivity, but in samples with elevated level of debris, the algorithm performed poorly, in this case the integration of relational features can improve the algorithm performance.

The integration of the relational features is implement through ***a selective-permeable filter***, a filter that discards some candidate cells, allow other candidate cells to go through the regular pipeline, and allow others to skip the intermediate filters in the pipeline and move forward to the highest, most sensitive layer of the pipeline (in this method the most sensitive layer is the visual word classifier).

7.5 Selective permeable relational features filter

Filters usually provide a binary response, either an object is allowed or not allow to move to the next step (permeable or impermeable). Our selective permeable filter provides a third option to skip some layers of the filter but still allow the object to

proceed to the final layer (Figure 7.6, 7.7). In this case, using relational features, the filter makes the following decisions:

a) Proceed: The candidate cells can proceed with the next layer of filtering.

b) Discard: The candidate cells in this category are very suspicious to be spurious, therefore are eliminated without further analysis. Since there is no turning back from this step, the classifier parameters for this category are extremely exclusive and is expected that only few cells fall in this group. In the relational streak filter the discard option includes only parallel streaks (cells that are close together and with similar height). To avoid eliminating true cells, the discard option is the last step in the relational filter. Under some conditions some parallel cells can be protected under the defer option in the lower layers of the filter.

c) Defer: This is the selective part of the filter, the candidate cells in this category are suspicions to be spurious and will probably be discarded in further layers of the filter but considering the spatial information and the relational characteristic of the cell, the filter defers the classification decision directly to the highest layer of the filter, protecting these candidate cells from intermediate filtering layers. For example, since parallel cells are discarded in the last layer of the relational filter, some parallel cells can be deferred directly to the visual words classifier skipping the discard step of the relational classifier filter.

7.6 Relational features parameters

The parameters of the relational features such as minimal width/width ratio or the number of streaks allow for each edge partition, and other features used in the filter were determined adaptatively considering all the streaks in the same sample.

a) Parallel streaks: Streaks were considered parallel if their centroids are 50 pixels or less apart in the 'x' coordinate, and 15 pixels or less apart in the 'y' coordinate, in addition their height must not be more than 10 pixels difference. Most parallel candidate cells are considered spurious, but depending on certain characteristic of the cell, provisions to protect non-spurious candidate cells are implemented in the relational selective permeable filter.

b) W**idth/Width ratio (WWR)**: Ratio of the width of two streaks (cell occurrences) in consecutive frames that belong to the same candidate cell. The closer this ratio is to 1 the more similarity in the width of the streaks. Therefore, streaks with WWR lower than the 20^{th} quantile of the WWR of all the streaks in the sample or those located in the edge (were the most of artifacts can appear), or is in the edge but is the not the only cell in the edge partition or is not the only cell in the frame, must proceed through the regular filter pipeline otherwise they are deferred.

c) Single-streak cells(SSCs): The non-spurious cells are expected to appear in more than one frame therefore most of the non-spurious cells should be represented by more

than one streaks. SSCs are suspicious to be spurious cells, therefore to avoid discarding no-spurious SSCs, all the SCCs that are a single cell in the frame, and they are not in the edge or they are in the edge but is the only one in the partition, are deferred.

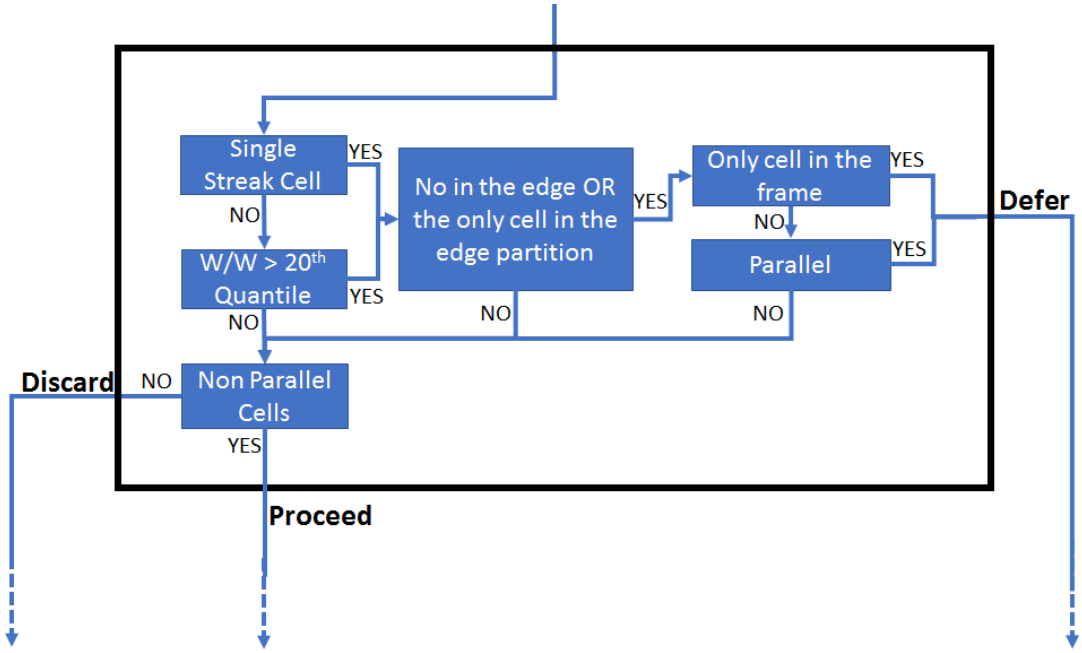


Figure 7.6: Selective Permeable Filter, Relational Feature Classifier. *First the cells are evaluated by the number of streaks. SSCs are further evaluated by their location and relation with other cells in the framer and in the sample to make the decision of proceed or defer. Non-SSCs in addition are evaluated according to their width/width ratio. The cells that are not deferred until this point are evaluated if they are parallel cells to decide to proceed or discard (no defer decision at this stage.*

7.7 Relational streak algorithm pipeline

The algorithm pipeline is composed of three major layers plus a classification decision protocol (Figure 7.7). The algorithm combines spatial and temporal information of the candidate cells along with relational features.

Layer I: Selective permeable filter also known as relational features classifier: Based on relational features, candidate cells are evaluated according to their location on the frame and in relation with other cells in the frame and/or the whole sample. Candidate cells at this layer can proceed to the next layer, can be discarded as spurious cells or can be defer to the highest sensitive layer (the visual words classifier), more detailed description of this layer is provided in figure 7.6.

Layer II: Geometrical & intensity distribution (GID) classifier: At this layer a machine learning routine is implemented to classify the candidate cells into the two main categories *'Te'* and *'Fe'* according to geometrical and intensity distribution features. These features were extracted in previous steps (as described above) and are feed in to the machine learning routine. The machine learning methods for this project were selected among the available methods provided by MATLAB "Classification Learner" App; these methods include: quadratic support vector machine, cubic support vector machine, median gaussian support vector machine, bagged tree ensemble, boosted tree ensemble, RUSBoosted tree ensemble, coarse decision trees, and fine K-Nearest Neighbors (KNN). These methods yielded the best performance on the F1score and sensitivity metric (F1 score over 0.8 sensitivity over 70%) for the classification of candidate cells in the 1 cell per ml dataset. At this layer all the candidate cells marked as "Procced" by the first layer are classified into the two main categories. The candidate cells marked as 'Defer' skip this layer and classified in the third layer.

Layer III: Visual words classifier, bag of features classifier (BoF). This method was discussed in the chapter 3, its implementation in this project was discussed above. Briefly, feature point locations are identified in each image fragment and SURF features are extracted from the selected point locations providing a descriptor for each interest point. Then, the features are clustered in groups according to their descriptor, each cluster becomes a visual word in the visual vocabulary. MATLAB uses the K-Means method to cluster the features to create a 500-word vocabulary. Then, a histogram is computed using the visual words, thus the images of occurrences of candidate cells can be represented as a histogram according to the frequency of their visual words. At this layer all the candidate cell proceeding from the second layer along with those marked deferred are classified into the two major categories 'Te' and 'Fe'.

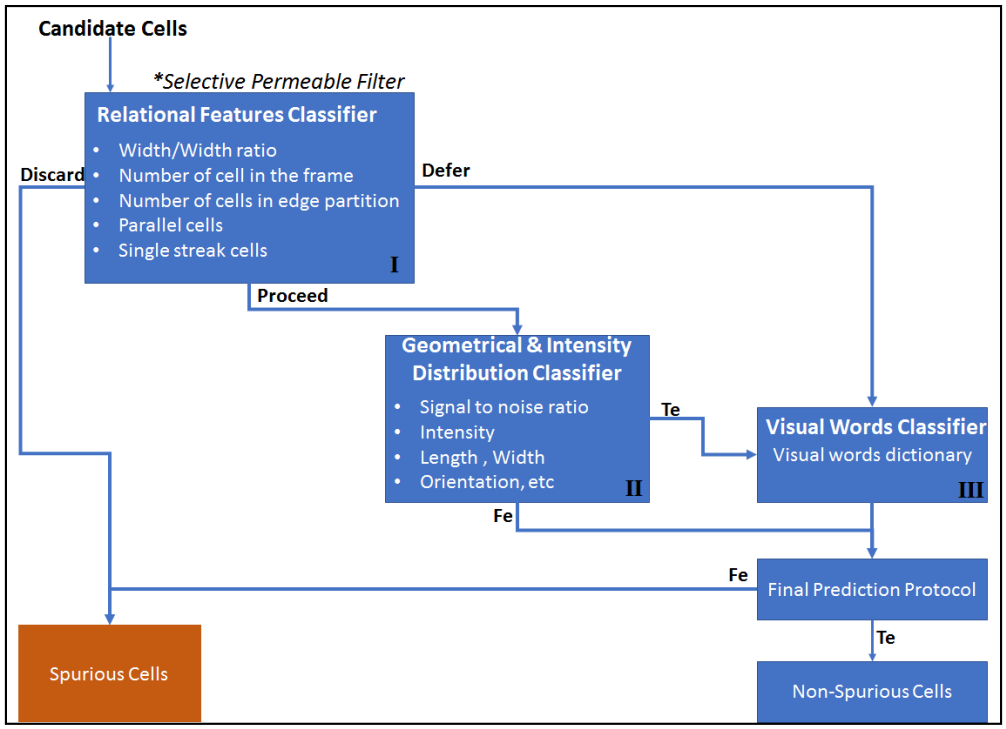


Figure 7.7: Classification pipeline. *Layers 1 take into relational features of the cells to decide on proceed or defer. Layers 2 and 3 made the classification decisions. All deferred cells skip all the intermediate layers and proceed directly to the last layer for classification decisions.*

7.8 Final prediction protocol

At this point the algorithm has provided two set of classifications, by the GID layer and by the BoF classifier. The function of the final prediction protocol is to combine these classifications to provide a final classification; in addition, it is also charged to solve classification conflicts. The final classification protocol follows the following steps:

a) By default, all the streaks are originally classified as '*Fe'*, this includes streaks from the cells discarded by the selective permeable filter.

b) All streaks classified as '*Fe'* by the GID are send to the final prediction protocol.

c) All the streaks classified as '*Te'* by the GID classifier are reclassified by the BoF classifier, either as '*Te'* or '*Fe'*, and send to the final prediction protocol.

d) All the steaks marked as "Defer" are classified according to the BoF classifier, either as '*Te'* or '*Fe'* and send to the final prediction protocol.

By the end of step 4, all the candidate cell occurrences have a classification either as '*Te'* or '*Fe'*.

Classification conflicts: Non-SSCs can generate a classification conflict when one occurrence is classified differently that another occurrence that represent the same candidate cell. This may happen when one occurrence is marked 'Defer' while the other occurrence is classified *'Fe'* by the GID classifier. In this case the deferred occurrence can be classified as *'Te'* by the BoF classifier generating a conflict. A classification conflict can also happen if the BoF classify two occurrences from the same candidate cell differently. In this case, if the candidate cell has odd number of occurrences, the conflict is solved by simple majority, otherwise (in case that the candidate cell have even number of occurrences) the conflict is solved with positive selection e.g. the candidate cell is classified as *'Te'*. Figure 7.8(a) and (b) represent the final decision protocol and the classification conflict resolution. Figure 7.8(a) represent a candidate cell with two streaks occurrence, one of the occurrences was deferred by the relational filter (Layer I), skipping the GID classifier (Layer II), and later is classified by the BoF classifier as *'Te'* (layer III), the occurrence is placed in the final prediction protocol as *'Te'*. On the other hand, the second occurrence from the same candidate cell is classified as *'Fe'* by the GID classifier and placed in the prediction protocol as *'Fe'* generating a conflict. In this case the conflict is solved by favoring the positive selection and both occurrences are classified as *'Te'*.

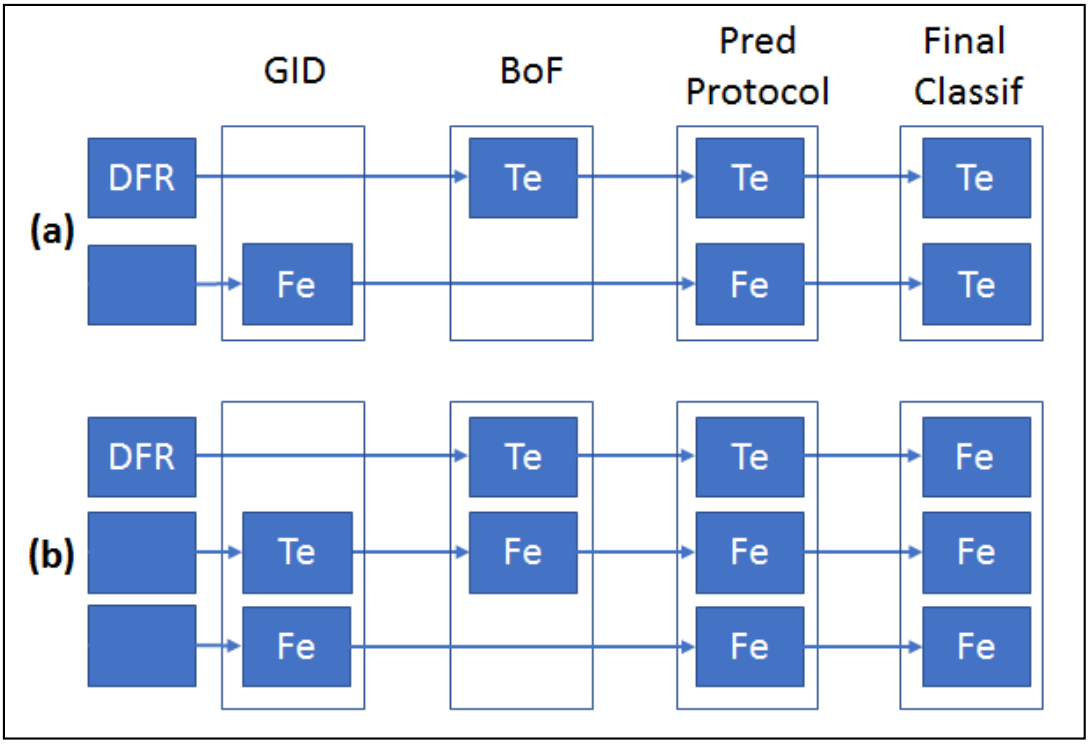


Figure 7.8: Final prediction protocol and classification conflict resolution. *(a) A candidate cell with two streaks occurrence. (b) A candidate cell with three streaks occurrence.*

Figure 7.8(b) represent a candidate cell with odd number of occurrences. One of the streak occurrence is deferred by the relational filter and then classified as *'Te'* by the BoF classifier this case the conflict is solved by simple majority. The second streak occurrence was originally classified as *'Te'* by the GID classifier, but later reclassified as *'Fe'* by the BoF classifier generating a conflict. The third occurrence is classified as *'Fe'* by the GID classifier. In this case the conflict is solved by simple majority.

Chapter 8: Experimental Results

8.1 Geometrical & intensity distribution classifier

Eight machine learning method were selected from the MATLAB Learner Classifier App to implement the GID classifier. MATLAB provides the capability of automatically generate codes for the learning model selected. The selected models were exported from the Learner Classifier App into the streak algorithm to compare the performance of different learning models with the visual words classifier. The machine learning method selected for this work are the following:

quadratic support vector machine, cubic support vector machine, median gaussian support vector

machine, bagged tree ensemble, boosted tree ensemble, RUSBoost tree ensemble, coarse decision trees, and fine K-Nearest Neighbors (KNN). These methods yielded the best performance on the F1 score and sensitivity metric (F1 score over 0.8 sensitivity over 70%) for the classification of candidate cells in the 1 cell per ml dataset.

8.2 Relational features classifier

To improve the performance of the streak algorithm, relational features were incorporated in a selective permeable filter providing the option to defer the classification of some streaks to the highest sensitive classifier (in this case the visual word classifier) skipping intermediate levels of filtering.

8.3 Visual words classifier (bag of features)

The image dataset formed from all the occurrences of candidate cells for the training and validation of the classifiers were created as described in the previous section. The images for the 1 cell per ml and 1 cell per 10 ml dataset were processed separately and stored in different folders.

As mentioned previously, the SURF features were extracted from the image data using the MATLAB function *"bagOfFeatures()"*. The SURF features are stored as a *"bagofFeatures"* object and later used to train the image classifier using the MATLAB function *"trainImageCategoryClassifier()"*. This function take as input the *"bagofFeatures"* object (created in the previous step) and an image dataset (the training set) and create a category classifier model as show in the code below.

"categoryClassifier = trainImageCategoryClassifier(traininSet, bagofFeatueObject)"

The category classifier is stored as an *"imageCategoryClassifier"* object and used to predict the category labels in the validation set.

8.4 Cross validation

The images of the two datasets (1 cell per ml and 1 cell per 10 ml) were processed separately. The samples were randomly partitioned into five disjoint sets of

approximately equal size. A group of four partition sets of images was used as the "training set" to build the classification model (derived from the MATLAB "bagOfFeatures()" function). Then, using this classification model, the labels of the validation set (the remaining 5th partition set) were predicted into one of the two categories *(Fe, Te)*. The steps for the 5-folds cross validation procedure are described in more details below.

8.4.1 Sample partition

The index for the cross validation, were obtained through a customized routine *"crosvalindSmpl()"* based on the MATLAB function *"crossvalind()"* that randomly assigns each sample to one of the 5 partition sets. The output of *"crosvalindSmpl()"* is a table with the sample number paired with the index for one the 5 partition sets. Figure 8.1 shows the indexes for the sample partition for the 1 cell per ml dataset. The indexes for the cross validation were saved for further comparison among different machine learning procedures.

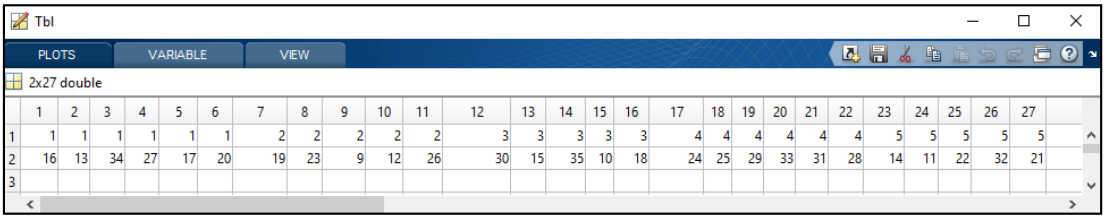


Figure 8.1: Output of "crosvalindSmpl()". *Each sample is randomly assigned to one of five groups, for example group 1 is formed by samples 16,13,34,27,17 and 20. In the first iteration of the 5-fold cross validation, samples from group 2,3,4 and5 are used as the training set and samples for group 1 are used as the validation set. In the second iteration samples from group 1,3,4, 5 are used as training set and samples from group 2, e.g. samples 19,23,9,12,26 are used as validation set and so on.*

8.4.2 Training and validation set

In each iteration (using the group index), the samples from 4 partitions are used as a training set and the samples from the remaining partition is used as the validation set. For example, in the first iteration samples from partitions 2, 3, 4 and 5 are used as the training set and samples from partition 1 (e.g. samples 16, 13, 34, 27, 17, and 20) are used as the validation set. In the second iteration samples from partitions 1, 3, 4 and 5 are used as training set and samples from partition 2, (e.g. samples 19,23,9,12 and 26) are used as validation set and so on (Figure 8.1).

8.4.3 Balancing the training and the validation set

After the training set is selected, the number of images in each category (*Te, Fe*) is balanced according to the category with the least number of images, using the MATLAB function *"splitEachLabel()"* as shows in the code line below:

"traininSet = splitEachLabel (TrainingSet, minSetCount, 'randomize')",

The code line above indicates that images from the category with the larger number are randomly selected to match the number of images in the category with the least number of images. For example, in the case of 1 cell per ml dataset, the category *'Te'* has 1077 images from a total of 25679 (category *Fe* has 24602 images), therefore 1077 images are randomly selected for category *"Fe"* to match the number of images from the category *'Te'*, leading to a balanced training dataset.

8.4.4 Classification prediction

Using the category classifier in each iteration, the non-spurious candidate cells in the samples corresponding to the 5th partition are classified into the two categories (*Fe, Te*) and aggregated in table for analysis of the classifier performance. The same procedure is performed for both sample datasets (1 cell per ml and 1 cell per10 ml dataset).

8.5 Results

Since the algorithm combine two classifiers; the geometrical & intensity distribution (GID) classifier and the visual words classifier (bag of features, BoF), a series of tests were designed to evaluate the performance of these two classifiers as standalone methods compared with the relational streak algorithm. Recall that the relational streak algorithm combines GID, visual words and relational features for classification of the candidate cells. The GID classifier was implemented using the machine learning routines mentioned above, each one evaluated as standalone method. For the visual words classifier (bag of features) MATLAB relies on the multiclass linear SVM as the default classification routine. For comparison between the different methods, the same cross-validation partition of the samples was used for these tests. The results are shown in Tables 1 - 4.

Table 8.1 shows the result of performance of the two classifiers as standalone methods. The GID classifier using quadratic SVM yield the highest F1 score (0.91). The BoF

classifier yields the highest sensitivity but produces the third highest number of false positives diminishing its F1 score (0.88). In general, all methods performed relatively well with average sensitivity of 85% and average F1 score of 0.87.

Table 8.1: Performance of the GID and BoF classifiers (1 cell/ml) as standalone methods. FP = False Positive; FN= False Negative; TP = True Positive; GT =Ground Truth

Classifier	Method	FP	FN	TP	GT	Sensitivity	Precision	F1 score
Visual Words Classifier	Visual Words (bag of features)	109	73	668	741	0.90	0.87	0.88
Geometrical & Intensity Distribution Classsifier	Bagged Tree Ensemble	28	115	626	741	0.85	0.96	0.89
	Coarse Decision Tree	30	168	573	741	0.78	0.95	0.85
	Cubic SVM	57	100	641	741	0.87	0.93	0.89
	Medium Gaussian SVM	23	113	628	741	0.85	0.97	0.90
	Quadratic SVM	18	104	637	741	0.86	0.97	0.91
	Boosted Tree Ensemble	32	104	637	741	0.86	0.96	0.90
	Fine KNN	145	135	606	741	0.82	0.83	0.81
	RUSBoosted Tree Ensemble	237	80	661	741	0.89	0.74	0.81
Average						**0.85**	**0.91**	**0.87**

Table 8.2 shows the performance of the GID classifier and the BoF classifier as standalone methods for the 1 cell per 10 ml dataset. The visual words method shows the highest sensitivity but the second lowest F1 score (0.27). This is due to considerable number of false positives with consequent false discovery rate. The best F1 score was obtained by the GID classifiers using Bagged Tree Ensemble as the classification method. In general, the methods did not perform as well in the 1 cell per 10 ml dataset as in the 1 cell per ml dataset. The average sensitivity for the 1 cell per 10 ml dataset was 62% with a F1 score of 0.49.

Table 8.2: Performance of the GID and BoF classifiers (1 cell/10ml) as standalone methods. FP = False Positive; FN= False Negative; TP = True Positive; GT =Ground Truth

Classifier	Method	FP	FN	TP	GT	Sensitivity	Precision	F1 score
Visual Words Classifier	Visual Words (bag of features)	229	7	47	54	0.84	0.17	0.27
Geometrical & Intensity Distribution Classsifier	Bagged Tree Ensemble	2	27	27	54	0.58	0.82	0.64
	Coarse Decision Tree	10	30	24	54	0.53	0.76	0.58
	Cubic SVM	31	23	31	54	0.61	0.57	0.51
	Medium Gaussian SVM	4	28	26	54	0.55	0.73	0.59
	Quadratic SVM	12	28	26	54	0.55	0.65	0.50
	Boosted Tree Ensemble	11	25	29	54	0.59	0.73	0.63
	Fine KNN	28	26	28	54	0.57	0.51	0.45
	RUSBoosted Tree Ensemble	396	12	42	54	0.76	0.15	0.23
Average						**0.62**	**0.57**	**0.49**

Table 8.3 shows the performance of the relational streak algorithm. The relational streak algorithm shows similar average sensitivity as the standalone methods but produced a better average F1 score. The average sensitivity for the relational feature algorithm is 87% (85% for the standalone methods) and an average F1 score of 0.91 (0.87 for the standalone methods).

Table 8.3: Performance of the Relational Streak algorithm (1cell/ml). FP = False Positive; FN= False Negative; TP = True Positive; GT =Ground Truth

Method	FP	FN	TP	GT	Sensitivity	Precision	F1 score
Bagged Tree Ensemble	39	94	647	741	0.87	0.95	0.90
Coarse Decision Tree	37	120	621	741	0.84	0.95	0.89
Cubic SVM	32	87	654	741	0.88	0.96	0.91
Medium Gaussian SVM	26	94	647	741	0.87	0.96	0.91
Quadratic SVM	39	86	655	741	0.89	0.95	0.91
Boosted Tree Ensemble	33	93	648	741	0.88	0.95	0.91
Fine KNN	38	103	638	741	0.86	0.95	0.90
RUSBoosted Tree Ensemble	33	75	666	741	0.90	0.95	0.92
Average					**0.87**	**0.95**	**0.91**

Table 8.4 shows the performance of the Relational Streak algorithm for the 1 cell per 10 ml dataset. The performance of the algorithm showed considerable improvement over the performance of the standalone methods. The average sensitivity for the relational algorithm is 82% with a F1 score of 0.73 compared with 62% and 0.49 for the standalone methods respectively.

Table 8.4: Performance of the Relational Streak algorithm (1 cell/10 ml). FP = False Positive; FN= False Negative; TP = True Positive; GT =Ground Truth

Method	FP	FN	TP	GT	Sensitivity	Precision	F1
Bagged Tree Ensemble	14	9	45	54	0.80	0.73	0.71
Coarse Decision Tree	13	10	44	54	0.79	0.79	0.76
Cubic SVM	22	7	47	54	0.84	0.68	0.73
Median Gaussian SVM	11	8	46	54	0.82	0.81	0.80
Quadratic SVM	23	8	46	54	0.82	0.71	0.70
Boosted Tree Ensemble	12	7	47	54	0.84	0.80	0.80
Fine KNN	14	7	47	54	0.84	0.77	0.75
RUSBoosted Tree Ensemble	51	8	46	54	0.82	0.61	0.64
Average					**0.82**	**0.74**	**0.73**

Up to this point the cross validation was performed using the same index for sample partitions, for the next tests, the samples are repartitioned each time (i.e. the partition indices of the samples are randomly selected each time).

In the next tests, the relational streak algorithm is compared with the its previous version as well as two standard cell tracking computational tools, MTrack2 and CellTrack.

The GID classifier for this test was selected from the eight classifiers tested before, in this case the method selected is bagged tree ensemble. Since, as discussed before, the previous version of the algorithm showed limited performance in samples with SNR lower than 4.41 dB, the results of this tests are ordered according to average SNR (Table 8.5).

Table 8.5: Results of the Relational Streak algorithm (1 cell/m). The samples are ordered by average SNR. FP = False Positive; FN= False Negative; TP = True Positive; GT =Ground Truth.

ID	FP	FN	TP	GT	Sensitivity	Precision	F1 score	SNR
27	1	5	15	20	0.75	0.94	0.83	2.86
28	0	7	23	30	0.77	1.00	0.87	3.03
35	0	9	21	30	0.70	1.00	0.82	3.41
34	1	12	17	29	0.59	0.94	0.72	3.42
32	2	4	19	23	0.83	0.90	0.86	3.50
31	2	6	23	29	0.79	0.92	0.85	3.55
29	0	5	25	30	0.83	1.00	0.91	3.71
23	2	3	31	34	0.91	0.94	0.93	4.05
26	0	4	32	36	0.89	1.00	0.94	4.08
30	4	4	21	25	0.84	0.84	0.84	4.15
33	2	7	18	25	0.72	0.90	0.80	4.27
24	0	4	25	29	0.86	1.00	0.93	4.41
17	2	3	21	24	0.88	0.91	0.89	4.52
25	2	5	29	34	0.85	0.94	0.89	4.56
22	8	1	26	27	0.96	0.76	0.85	4.58
15	2	1	22	23	0.96	0.92	0.94	4.64
20	2	1	30	31	0.97	0.94	0.95	5.02
21	0	0	21	21	1.00	1.00	1.00	5.08
19	1	2	27	29	0.93	0.96	0.95	5.09
18	1	2	29	31	0.94	0.97	0.95	5.37
13	1	2	28	30	0.93	0.97	0.95	5.91
12	1	1	20	21	0.95	0.95	0.95	6.02
16	1	0	21	21	1.00	0.95	0.98	6.20
14	0	1	30	31	0.97	1.00	0.98	6.25
11	1	1	25	26	0.96	0.96	0.96	6.46
9	1	1	26	27	0.96	0.96	0.96	7.42
10	2	1	24	25	0.96	0.92	0.94	7.44

In samples with SNR equal or greater than 4.41 dB, the streak algorithm and the relational streak algorithm performed similarly (Table 8.6). In samples with SNR lower than 4.41 dB, the relational streak algorithm shows considerable improvement over the previous version with an average F1 score of 0.85 and average sensitivity of 78%. The relational streak algorithm significantly outperforms MTrack2 and CellTrack for all SNRs.

Table 8.6: Performance of the relational streak detection algorithms (1 cell/ml) dataset compared with other cell tracking methods. The table show the samples clustered by SNR groups

	SNR< 4.41 db			SNR >= 4.41 dB			All SNR		
	Sensitivity	Precision	F1 score	Sensitivity	Precision	F1 score	Sensitivity	Precision	F1 score
CellTrack	0.26	0.19	0.22	0.60	0.29	0.38	0.46	0.25	0.31
MTrack2	0.43	0.27	0.30	0.71	0.80	0.74	0.59	0.58	0.56
Streak Algorithm	0.58	0.88	0.69	0.91	0.93	0.92	0.78	0.91	0.83
Relational Streak Algorithm	0.78	0.94	0.85	0.94	0.94	0.94	0.88	0.94	0.91

Table 8.7 shows the results of the relational streak algorithm for samples with nominal concentration of 1 cell per 10 ml. For this test the GID classifier was implemented using bagged tree ensemble for classification, the indexes of the sample were randomly selected. As in the 1 cell per ml dataset, the samples in this test are ordered by average SNR.

Table 8.7: Results of the relational algorithm (1 cell/10 ml). Samples are ordered by SNR. FP = False Positive; FN= False Negative; TP = True Positive; GT =Ground Truth.

ID	FP	FN	TP	GT	Sensitivity	Precision	F1 score	SNR
12	1	0	4	4	1.00	0.80	0.89	1.64
7	0	0	4	4	1.00	1.00	1.00	1.84
15	0	1	0	1	0.00	0.00	0.00	1.99
6	0	1	3	4	0.75	1.00	0.86	3.07
11	1	0	6	6	1.00	0.86	0.92	3.16
8	1	2	1	3	0.33	0.50	0.40	3.24
14	1	0	2	2	1.00	0.67	0.80	3.30
4	0	2	7	9	0.78	1.00	0.88	3.57
13	2	0	7	7	1.00	0.78	0.88	3.72
10	0	0	3	3	1.00	1.00	1.00	4.04
2	0	0	3	3	1.00	1.00	1.00	5.27
3	0	1	2	3	0.67	1.00	0.80	5.94
5	1	0	3	3	1.00	0.75	0.86	6.00
1	0	0	2	2	1.00	1.00	1.00	7.69
9	1	0	0	0	1.00	0.00	0.00	N/A

Table 8.8 shows the results of the performance of the relational streak algorithm using the 1 cell per 10 ml dataset. In this dataset, the previous version of the algorithm showed limited performance in all levels of SNR. The relational streak algorithm showed improvements in the F1 score in all levels of SNR. Both algorithms showed similar sensitivity for samples with average SNR equal or greater than 4.41 dB, but the relational algorithm yielded a much better F1 score. The relational streak algorithm performed better than the current cell tracking method used in this test.

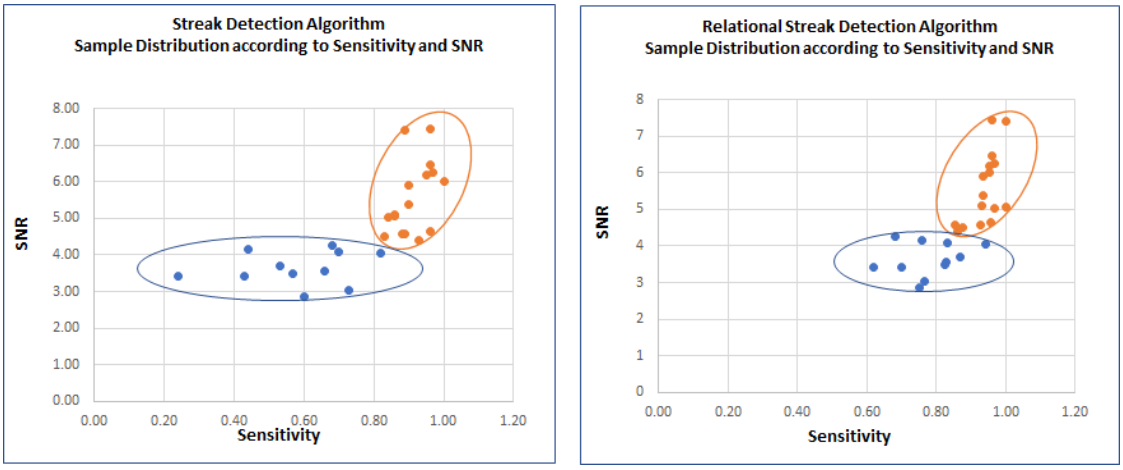
Table 8.8: Performance of the relational algorithms (1cell/10 ml) compared with other cell tracking methods. The table show the samples clustered by SNR groups

Method	SNR< 4.41 db			SNR >= 4.41 dB			All SNR		
	Sensitivity	Precision	F1 score	Sensitivity	Precision	F1 score	Sensitivity	Precision	F1 score
CellTrack	0.37	0.01	0.02	0.46	0.00	0.00	0.39	0.01	0.01
MTrack2	0.37	0.18	0.12	0.63	0.59	0.55	0.44	0.30	0.27
Streak Algorithm	0.52	0.37	0.38	0.92	0.49	0.60	0.65	0.38	0.41
Relational Streak Algorithm	0.79	0.76	0.76	0.92	0.94	0.91	0.84	0.76	0.75

Chapter 9: Discussion and Conclusion

9.1 Analysis of results, 1cell per ml dataset

Preliminary data from the previous version of the algorithm indicated that the algorithm performed well in samples with SNR equal or greater than 4.41 dB but performed poorly in samples with SNR lower than 4.41 dB in the 1 cell per ml dataset. One of the motivations of this work was to develop an algorithm able to detect cells regardless of the level of noise. For this purpose, we evaluated the correlation between sensitivity and F1 score with SNR values (clustering high SNR and low SNR values) in the two versions of the algorithm e.g. streak detection v/s relational streak detection algorithm. Comparing sensitivity vs SNR in the two algorithms in samples with SNR lower that 4.41 dB, the sensitivity of the relational algorithm ranges from 0.62 to 0.94, but the streak algorithm shows a more spread range between 0.24 to 0.82 (Figure 9.1).



a *b*

Figure 9.1: Clustering of the samples with low and high SNR*. (a) Streak detection algorithm. (b) Relational streak algorithm. The streak algorithm show a more spread range of sensitivity than the relational algorithm*

The coefficient of variation (CV) in this group (SNR<4.41) was considerable different in the two algorithms (0.12 and 0.28 for the for the relational and the streak algorithm respectively) indicating that the sensitivity shows greater variability in the streak algorithm than the relational algorithm for samples with low SNR (Table 9.1).

Table 9.1: Performance of the streak v/s relational algorithm (1 cell/ml) for samples with low and high SNR. The sensitivity of the streak algorithm in samples with low SNR shows a greater coefficient of variation than the relational algorithm (0.28 and 0,12)

	SNR <4.41			SNR >= 4.41		
	Standar deviation	Mean	Coeficient of variation	Standar deviation	Mean	Coeficient of variation
Streak detection algorithm	0.16	0.58	0.28	0.05	0.91	0.06
Relational streak algorithm	0.09	0.78	0.12	0.04	0.94	0.05

In addition, a t-test (two tails, type 1) shows significant difference between the sensitivity of the relational algorithm compared with the streak algorithm ($p=0.003$) in samples with low SNR. The F1 score of the two algorithms also shows significant difference ($p=0.001$) in this SNR group (Table 9.2). The precision however did not show significance difference between the two algorithms in this SNR group ($p=0.173$). This is due to the proportion of false positives and false negatives. Precision does not consider false negatives; conversely, sensitivity does consider false positives.

Table 9.2: t-test result comparing sensitivity, precision and F1 score (1 cell/ml) for the streak algorithm and the relational algorithm for samples with low and high SNR.

Method	SNR< 4.41 dB			SNR >= 4.41 dB			All SNR		
	Sensitivity	Precision	F1 score	Sensitivity	Precision	F1 score	Sensitivity	Precision	F1 score
Streak Detection Algorithm	0.58	0.88	0.69	0.91	0.93	0.92	0.78	0.91	0.83
Relational Streak Algorithm	0.78	0.94	0.85	0.94	0.94	0.94	0.88	0.94	0.91
t-test, p value	0.0003	0.173	0.001	0.048	0.500	0.070	0.0002	0.123	0.001
Wilcoxon Rank Sum test	0.002	0.232	0.002	0.102	0.663	0.056	0.062	0.214	0.024

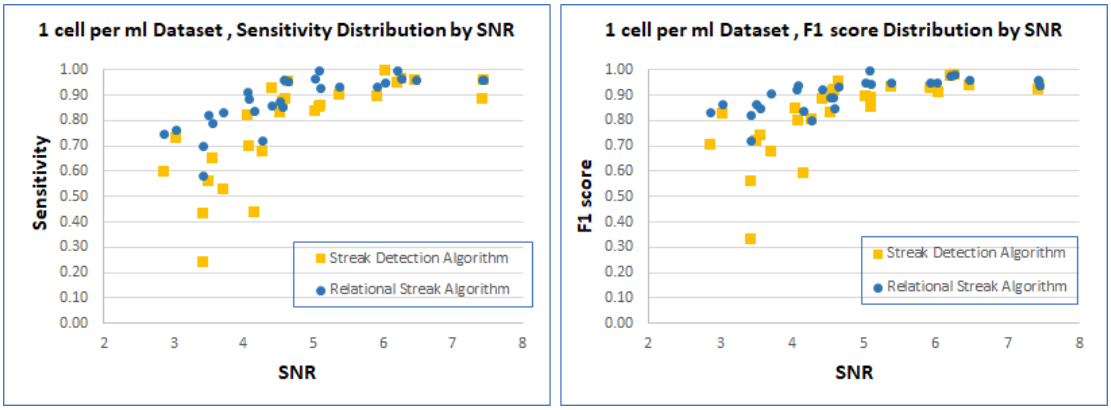
Table 9.3 shows a summary of the false positive and false negative results from both algorithms. The streak detection algorithm has a disproportional large number of false negatives compared with the relational streak algorithm (128 and 66 respectively). A t-test shows that there is significance difference between false negatives in both algorithms *(p=0.0003)*, but did not show significance difference between the false positive results *(p=0.27)*.

Table 9.3: Streak detection v/s Relational algorithm (1 cell/ml). False positive and false negative numbers for the streak detection algorithm and the relational streak algorithm on samples with SNR lower than 4.41 dB.

ID	Number of False Positives		Number of False Negatives		SNR
	Streak detection Algorithm	Relational streak algorithm	Streak detection Algorithm	Relational streak algorithm	
27	2	1	8	5	2.86
28	1	0	8	7	3.03
35	3	0	17	9	3.41
34	6	1	22	12	3.42
32	0	2	10	4	3.50
31	3	2	10	6	3.55
29	1	0	14	5	3.71
23	4	2	6	3	4.05
26	2	0	11	4	4.08
30	1	4	14	4	4.15
33	0	2	8	7	4.27
Total	**23**	**14**	**128**	**66**	
t-test, *p value*	**0.2767**		**0.0003**		
Wilcoxon Rank Sum test	**0.2959**		**0.0022**		

On the other hand, precision and F1 score, show no significance difference between the two algorithms for samples with SNR greater or equal to 4.41 dB (*p=0.5 and p=0.07* respectively), as shown in Table 9.2 above.

The data showed up to this point indicate that the relational algorithm performed better than the streak algorithm, but we have not answer the question if the algorithm is robust to noise. Figure 9.2 shows the sensitivity and F1 score distribution of the samples with nominal concentration of 1 cell per ml according to their average SNR values. In both cases the relational algorithm shows less spread and higher values of sensitivity and F1 score than the streak algorithm.



a *b*

Figure 9.2: Sensitivity and F1 score distribution according to SNR (1 cell/ml). *The relational streak algorithm shows less spread and better performance than the streak algorithm for samples with nominal concentration of 1 cell per ml. (a)Sensitivity distribution according to SNR. (b) F1 score distribution.*

To evaluate the robustness of the algorithm to changes in signal intensity and noise level, an intra-algorithm comparison of sensitivity, precision and F1 score between samples with low SNR and high SNR was performed. The results are showed in Table 9.4. The average sensitivity of the relational algorithm is 78% and 94% for samples with low and high SNR respectively, this value shows less spread than the average sensitivity for the streak algorithm for low and high SNR (58% and 91%), however a t-test (two tails , type 3) shows significant statistical difference between the average

sensitivity for the relational streak algorithm between the samples with high and low SNR (*p=0.00014)* indicating that the algorithm is affected to noise.

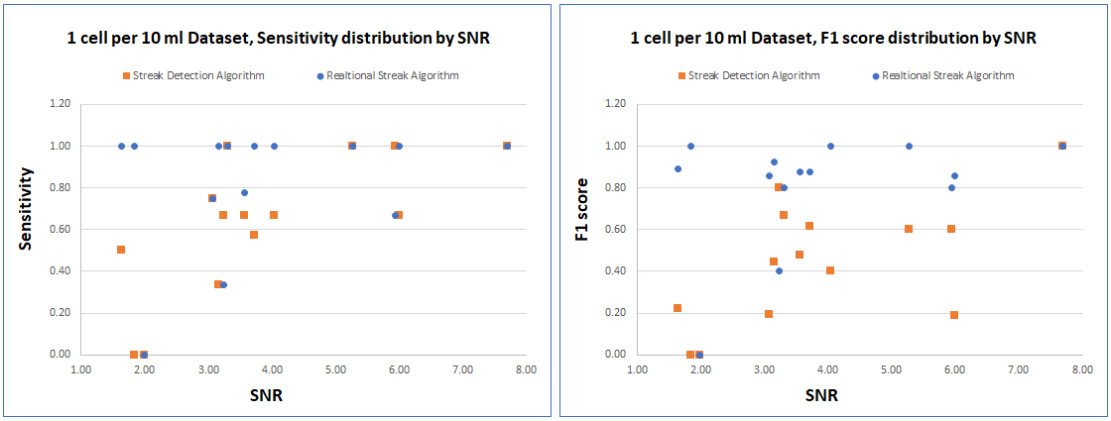
Comparing the F1 score for the relational streak algorithm, a t-test also shows significant statistical difference between samples with low and high SNR (*p=0.00058),* confirming the conclusion that the algorithm although improved its performance compared with the previous version, is still affected by noise and performed significantly better in samples with high SNR. than in samples with low SNR.

Table 9.4: Summary of the result (1 cell/ml). Sensitivity, precision and F1 score for the streak detection algorithm and the relational streak algorithm for the 1 cell per ml dataset.

SNR	Streak detection algorithm			Relational streak algorithm		
	Sensitivity	Precision	F1 score	Sensitivity	Precision	F1 score
SNR < 4.41	0.58	0.88	0.69	0.78	0.94	0.85
SNR >= 4.41	0.91	0.93	0.92	0.94	0.94	0.94
t-test, p value	0.00004	0.25105	0.00052	0.00014	0.97344	0.00058
Wilcoxon Rank Sum	0.00002	0.31069	0.00002	0.00005	0.67074	0.00027

9.2 Analysis of results, 1cell per 10 ml dataset

In preliminary result the streak algorithm performed poorly in samples with nominal concentration of 1 cell per 10 ml. The relational algorithm in general shows higher values of sensitivity and F1 score in all levels of SNR compared with the streak algorithms (Figure 9.3).



a *b*

Figure 9.3: Sensitivity and F1 score distribution according to SNR (1 cell/10ml). *The relational streak algorithm shows less spread in sensitivity and F1 score and better performance than the streak algorithm. (a)Sensitivity distribution according to SNR. (b) F1 score distribution.*

Comparison between the F1 scores of the two algorithms shows statistically significant difference (t-test, 2 tails, type 1) in samples with SNR lower than 4.41 dB (*p=0.015),* and for all SNR (*p=0.003).* Sensitivity improves from 0.52 to 0.79 in samples with SRN lower than 4.41 dB. Sensitivity also improves from 0.65 to 0.84 considering all samples regardless SNR values, however no statistically significant difference was found comparing the sensitivity of the two algorithms in these SNR groups (*p=0.057, p =1, and p = 0.075),* as showed in Table 9.5.

Table 9.5: Summary of the result (1 cell/10 ml). Sensitivity, precision and F1 score for the streak detection algorithm and the relational streak algorithm for the 1 cell per 10 ml dataset.

Method	SNR< 4.41 db			SNR >= 4.41 dB			All SNR		
	Sensitivity	Precision	F1 score	Sensitivity	Precision	F1 score	Sensitivity	Precision	F1 score
Streak Detection Algorithm	0.52	0.37	0.38	0.92	0.49	0.60	0.65	0.38	0.41
Relational Streak Algorithm	0.79	0.76	0.76	0.92	0.94	0.91	0.84	0.76	0.75
Total samples	10	10	10	4	4	4	15	15	15
t-test, p value	0.057	0.028	0.015	1.000	0.058	0.113	0.075	0.003	0.003
Power of t-test	0.662	0.913	0.976	0.050	0.391	0.262	0.522	0.980	0.976

In addition, like the 1 cell per ml dataset, no statistically significant difference was found in samples with SRN greater than or equal to 4.41 dB for sensitivity and F1 score (*p=1 and, p =0.15* respectively*)* for the 1 cell per 10 ml dataset.

These results (lack of statistical significance) could be attributed to the lack of power of the t-test to reject the false null hypothesis (type II error) due to the small sample size and effect size. Using the MATLAB function *"sampsizepwr()"*, we find that for samples with SNR equal or greater than 4.41 dB the power of the t-test is less than 40% for the three metrics e.g. sensitivity, precision and F1 score (Table 9.5) indicating that there is a 60% chance of a type II error. This is mostly due to the small sample size for this group (n = 4). Similarly, Table 9.5 also shows 66% and 52% power of the t-test on the sensitivity of the algorithm for samples with SNR lower than 4.41 and for all SNRs respectively. In these the number of samples seems appropriate, but the effect side is rather small, e.g. from 0.52 to 0.79 in samples with SNR lower than 4.41 dB, and from 0.65 to 0.84 in all SNRs.

9.3 Conclusion

In this dissertation, an algorithm for detection and tracking of rare cells for streak mode imaging in a wide-field flow cytometer (originally designed for low resources settings) was developed. The algorithm integrates geometrical and intensity distribution features of streak images with relational features and visual words for classification and counting of rare cells. The algorithm achieves sensitivity of 94% with a F1 score of

0.94 in samples with nominal concentration of 1 cell per ml, and 84% sensitivity with F1 score of 0.75 in samples with nominal concentration of 1 cell per 10 ml.

The algorithm improved performance from its previous versions in both datasets. In the 1 cell per 10 ml dataset, on samples with SNR lower than 4.41dB the algorithm's F1 score improved from 0.38 to 0.76, and in all SNRs the algorithm improved it's F1 score from 0.41 to 0.75. In this dataset, on samples with SNR equal or greater than 4.41 dB the algorithm also showed improvement from 0.60 to 0.91, however due to small sample size this finding need to be subject to further research. The algorithm also outperformed the two cell-tracking computational tools (MTrack2 and CellTrack) used in this work.

The algorithm is still affected by noise and performed better in samples with SNR equal or greater than 4,41 dB, but the Relational Streak algorithm showed more robustness to changes in intensity and noise level than its previous version. It is worth pointing out that even with low quality images the Relational Streak algorithm has the capability of detecting very low number of cells (e.g. nominal concentration of 1 cell per 10 ml) in a large volume (30 ml).

In conclusion, in this thesis, we have demonstrated the utility of the Relational Streak algorithm for the identification and accurate classification of rare cells from streak cytometry images. The main advantage of this technology is the unique capability of analyzing very small cell number (0.1 cell /ml) in a large volume (10 ml) in short time

(1 minute) enabling the analysis of rare cell in clinical samples. The Relational Streak algorithm is simple, intuitive and take advantage of the relational features of the objects for eliminating artifacts and accurate cell classification. The algorithm was implemented in MATLAB, thus heavily depending on dedicated MATLAB packages and libraries that involve license fees incompatible with low resources settings. MATLAB was selected only for prototyping purposes and is envisioned that equivalent codes can be developed in open source C++ libraries such as OpenCV for the future implementation of the algorithm in low resources settings. The wide-field steak mode flow cytometer empowered with the relational algorithm for cell detection and counting results in a simpler, affordable and portable flow cytometer which could facilitate the integration of cell-based diagnosis in low resources settings.

Chapter 10: Future Development and Potential Applications

10.1 Limitations and future development

As was mentioned before, the Relational Streak Algorithm performed better in samples with high signal to noise ratio (SNR > 4.41 dB) than samples with low SNR. The quality of the signal and the noise levels contributing to the low SNR can be attributed to thermal noise of the CCD compounded by the long exposure time. In future development of this technology, this issue could be addressed by using a higher quality imager (e.g., a cool CCD) with consequent improvement of the detection level[161]. Sample autofluorescence can also elevate noise. Sample autofluorescence arises from endogenous tissue components and fixative in human or animal samples and result in a broad emission that overlaps the fluorescence signal. One characteristic of autofluorescence is its complex intensity decay from multiple components with a range of short lifetimes (up to few ns). Therefore, one way to decrease the impact of autofluorescence is to use dyes with longer decay lifetimes than autofluorescence components. For example, azadioxatriangulenium (ADOTA) has a lifetime of ~25ns [162]. Rich, et al. showed that using ADOTA improves SNR by eliminating > 96% of autofluorescence discarding photons detected within 20ns of the excitation pulse [163].

The algorithm described here was implemented for a single wavelength fluorescence excitation and single wavelength emission limiting the detection to a single color for single biomarker. However, flow cytometry applications usually require multiple

biomarkers to allow interrogation and counts of distinct cell types at the same time. For example, individuals infected with immunodeficiency virus (HIV) type 1, usually have more CD8 than CD4 lymphocytes in their blood (CD4/CD8 ratio <1)[164], whereas healthy individuals have a CD4/CD8 ratio between 1 and 4, and people with immune diseases tend to have a higher CD4/CD8 ratio. Therefore, multiplexed detection would expand the capabilities of this technology for evaluation of cell population ratios for HIV diagnosis in LMICs. Since multiple targets can also be identified in the same cell, the efficiency of the algorithm can be improved by decreasing the false negative rate, e.g., if the signal is weak in one target, the cell can still be identified using the second target. Multiplexed detection can be achieved by using a single multiwavelength excitation source and a single camera with a split filter for two color detection, see figure 10.1(II). Another approach could use two cameras each equipped with a different filter, see figure 10.1(III)). For multiplexing the algorithm can be extended by running the streak detection process for each fluorescent marker independently and then eliminating the spurious cells by combining cells identified by the markers used. For best detection multiple excitation sources with a matching filter can be used (specific to the target fluorophore). However, this configuration would complicate the design of the device.

An alternative to the use filters for multi-maker detection is digital separation of colors into several spectral components (the image can be digitally sliced into distinct R/B/G channels) for analysis of emission intensity for each source, see figure 10.1(IV). Zhang, et al. proposed this filter-free approach to simultaneously identify green and red signals

in fluorescently labelled cells for development of an affordable miniature fluorescence microscope for point of care diagnosis [165,166]. Higher spectral resolution can be obtained by using multispectral analysis. Digital color separation can be combined with the color separation methods described above to improve spectral resolution.

To implement these approaches, the Relational Streak algorithm will need to be modified to incorporate the difference in the signal acquiring method. For example, the exposure and the flow rate must be adjusted to produce long streaks which can be detected by both parts of the filter. Our relational streak algorithm can be implemented using this technique to identify multiple colors and empower the streak cytometer with multiplexing capabilities.

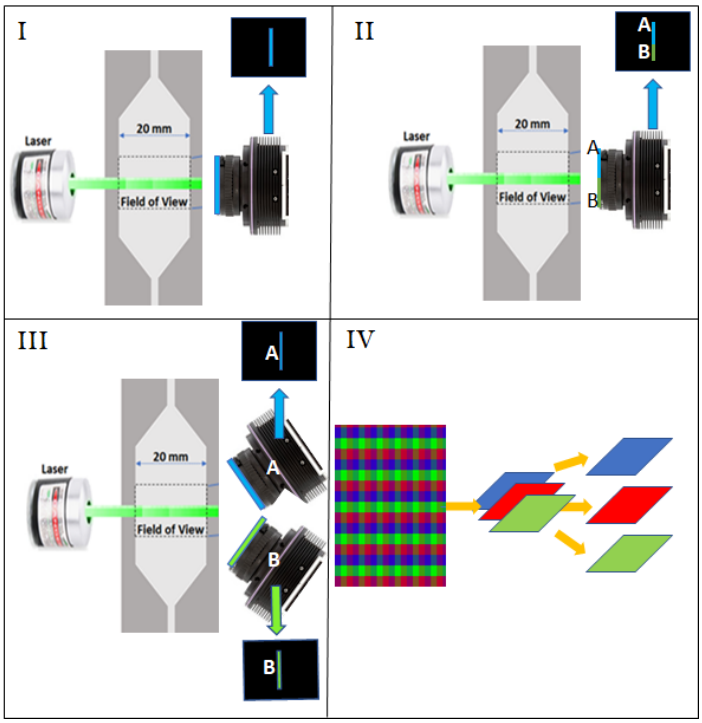


Figure 10.1: Multicolor streak flow cytometry. *I) monochrome detection (current method). II) One camera with split filter. III) Two cameras with two filters. IV) Digital color separation.*

Section10.2 Potential applications

The main advantage of this technology is its unique capacity for analyzing very small cell numbers (0.1 cell /ml) in a large volume (10 ml) in short time (1 minute). There are many rare-cell clinical applications besides circulating tumor cells (demonstrated in this dissertation) that can utilize this unique capability. In all of these applications, different cut-off values and parameters may need to be used, but the fundamental principle of integrating relational features, visual words, and geometrical/intensity distribution features for object detection still applies.

10.2.1 Rare cells analysis other than CTCs

A major challenge to understand disease progression is identification of the molecular changes in a specific immune cell population, but this is difficult where key molecular drivers are exceedingly rare among tremendous cellular diversity, such as is the case for antigen-specific memory B cells in the lymph node. T-cells also have considerable cellular diversity from rearrangements of T-cell receptors (TCR) during their development. This feature provides extreme flexibility in the immune system for adapting to changing conditions and reactions to diverse antigens [167,168]. Haematopoietic stem cells (HSC) are another example where rare cell detection technology can be useful. HSC are found in peripheral blood in a proportion of 1 cell per 10^6 cells, comparable to CTCs (1 cell per 10^7 cells) (Table 10.1). Since HSCs have the ability differentiate into any other kind of blood cell, they play roles in the treatment of cancer and other disorders of the blood or immune system. In addition HCS in

peripheral blood can also differentiate in other cell linages such as adipocytes, which are useful in tissue engineering applications [167].

Table 10.01: Examples and frequency of rare cell types[167]

Cell type	**Frequency**
Antigen-specific T cells	1 cell per 10^6 cells (in peripheral blood)
Circulating tumor cells	1 cell per 10^7 cells (in peripheral blood)
Endothelial progenitor cells	1 cell per 10^5 cells (in peripheral blood)
Hematopoietic stem cells	1 cell per 10^4 cells (in bone marrow)
Hematopoietic stem cells	1 cell per 10^6 cells (in peripheral blood)

High throughput rare cell detection technology can also apply to study of respiratory diseases. Respiratory diseases, including lung infection and chronic obstructive pulmonary disease (COPD) (excluding cancer) accounted for around 4.3 million deaths in 2016 (third cause of death after ischemic heart disease and stroke) worldwide[169]. Extensive studies in cell populations of mononuclear phagocytes (MNPs) has been done in mouse models, but the challenges of working with rare cells in inaccessible human tissue has impeded the translation of these findings to respiratory human disease [170]. One common challenge of these applications is the need to evaluate large volumes of sample in short times to ensure the recovery of diagnostically useful rare cells. Bulk cell population analysis will dilute the contribution of rare cells to the overall expression pattern, and as a result, cause the loss of unique molecular signatures. In addition, transcriptions from contaminant cells can be mistaken for rare cell signatures and lead to erroneous conclusions[168]

10.2.2 Public health applications in LMICs

Wide field streak flow cytometry empowered by the Relational Streak algorithm can be used for many health applications in LMICs, for example, to study bacterial resistance to antibiotics. The spread of bacterial resistance to antibiotics present a substantial challenge for control of infectious diseases and represents one of the biggest threats to low resource areas and global health[171]. Common infectious diseases usually can be treated with known effective antibiotics, but rapid emergence of resistant bacteria has the effect of making many common infectious diseases such pneumonia or salmonellosis harder to treat [171-173]. One way to address this issue is to test the bacterial susceptibility profile before initiation of treatment. This makes antimicrobial susceptibility testing (AST) critical for successful disease treatment [174,175]. Traditional AST methods such disk diffusion and gradient diffusion require complex infrastructure that usually delivers results 24 or more hours after sampling[176]. Rapid AST methods exist, but if available, are prohibitively expensive for many LMICs[177]. The streak-imaging cytometer empowered by the relational streak algorithm may be able to help if implemented to quantify the rate of microbial growth in response to antibiotic exposure. Bacteria incubated briefly (~ 30 minutes) with an antibiotic and followed by a fluorescence viability assay to differentiate dead vs. live bacteria, and the ratio of live/dead bacteria calculated (this application was included in a grant proposal submitted to NIH). The Relational Streak algorithm also can be coupled with other technologies suitable for LMIC use, such as miniature microscopy [165] to detect and evaluate disease related bacteria in contaminated water or food. Disease related bacterial contamination of drinking-water is a major public health problem. The burden

of drinking-water related diseases is high in many LMICs where diarrhea remains a leading cause of child death. Global access to drinking water is monitored by WHO and UNICEF [178]. Since testing for individual pathogens is impractical and expensive, the environmental protection agency (EPA) proposes a total count of coliform bacteria as an alternative standard for determining the bacterial safety of waters [179]. For this application a flow cell could be integrated with a miniature microscope and a rapid bacterial count implemented using the Relational Streak algorithm.

Furthermore, this technology could be implemented in mobile smart phones (which are more commonly available than webcams in low and middle-income countries). The computational capabilities of the phone could support the analysis of the streak images and connect to the internet to transmit the results to a physician for interpretation. Smart phones also can also access remote processing more powerful than a phone's limited computational power for more complex analysis. In this case the mobile phone would transmit only data to a more powerful platform for analysis and interpretation.

Bibliography

1. Giuliano M, Giordano A, Jackson S, et al. Circulating tumor cells as early predictors of metastatic spread in breast cancer patients with limited metastatic dissemination. *Breast Cancer Research*. 2014;16(5).
2. Alix-Panabieres C, Pantel K. Circulating Tumor Cells: Liquid Biopsy of Cancer. *Clinical Chemistry*. 2012;59(1):110-118.
3. Rejniak KA. Circulating Tumor Cells: When a Solid Tumor Meets a Fluid Microenvironment. *Advances in experimental medicine and biology*. 2016;936:93-106.
4. Zhang Z, Shiratsuchi H, Lin J, et al. Expansion of CTCs from early stage lung cancer patients using a microfluidic co-culture model. *Oncotarget*. 2014;5(23):12383-12397.
5. Chang C-L, Huang W, Jalal SI, et al. Circulating tumor cell detection using a parallel flow micro-aperture chip system. *Lab on a chip*. 2015;15(7):1677-1688.
6. Ferreira MM, Ramani VC, Jeffrey SS. Circulating tumor cell technologies. *Molecular Oncology*. 2016;10(3):374-394.
7. Millner LM, Linder MW, Valdes R, Jr. Circulating tumor cells: a review of present methods and the need to identify heterogeneous phenotypes. *Annals of clinical and laboratory science*. 2013;43(3):295-304.
8. Alvarez Cubero MJ, Lorente JA, Robles-Fernandez I, Rodriguez-Martinez A, Puche JL, Serrano MJ. Circulating Tumor Cells: Markers and Methodologies for Enrichment and Detection. *Methods in molecular biology (Clifton, NJ)*. 2017;1634:283-303.
9. Che J, Yu V, Dhar M, et al. Classification of large circulating tumor cells isolated with ultra-high throughput microfluidic Vortex technology. *Oncotarget*. 2016;7(11):12748-12760.
10. Organization WH. Cancer Fact Sheet. 2018; http://www.who.int/mediacentre/factsheets/fs297/en/.
11. Hay Burgess DC, Wasserman J, Dahl CA. Global health diagnostics. *Nature*. 2006;444 Suppl 1:1-2.
12. Rasooly R, Bruck HA, Balsam J, Prickril B, Ossandon M, Rasooly A. Improving the Sensitivity and Functionality of Mobile Webcam-Based Fluorescence Detectors for Point-of-Care Diagnostics in Global Health. *Diagnostics (Basel, Switzerland)*. 2016;6(2).
13. Herold KE, Rasooly A, eds. *Lab on a chip technology*. 2009/01/20 ed. Norfolk U.K.: Caister Academic press; 2009. Herold KE, Rasooly A, eds; No. 1.
14. Urdea M, Penny LA, Olmsted SS, et al. Requirements for high impact diagnostics in the developing world. *Nature*. 2006;444 Suppl 1:73-79.
15. Bollyky TJ, Templin T, Andridge C, Dieleman JL. Understanding The Relationships Between Noncommunicable Diseases, Unhealthy Lifestyles, And Country Wealth. *Health affairs*. 2015;34(9):1464-1471.

16. World Health Organization. Population Data by World Bank income group. 2015. Accessed 9 Dec, 2015.
17. Bank TW. Understanding Poverty. 2016; http://www.worldbank.org/en/topic/poverty/overview.
18. Sun S, Yang M, Kostov Y, Rasooly A. ELISA-LOC: lab-on-a-chip for enzyme-linked immunodetection. *Lab on a chip*. 2010;10(16):2093-2100.
19. Sapsford KE, Sun S, Francis J, Sharma S, Kostov Y, Rasooly A. A fluorescence detection platform using spatial electroluminescent excitation for measuring botulinum neurotoxin A activity. *Biosensors & bioelectronics*. 2008;24(4):618-625.
20. Sun S, Ossandon M, Kostov Y, Rasooly A. Lab-on-a-chip for botulinum neurotoxin a (BoNT-A) activity analysis. *Lab on a chip*. 2009;9(22):3275.
21. Sun S, Francis J, Sapsford KE, Kostov Y, Rasooly A. Multi-wavelength spatial LED illumination based detector for in vitro detection of botulinum neurotoxin A activity. *Sensors and Actuators B: Chemical*. 2010;146(1):297-306.
22. Kostov Y, Sergeev N, Wilson S, Herold KE, Rasooly A. A simple portable electroluminescence illumination-based CCD detector. *Methods in molecular biology (Clifton, NJ)*. 2009;503:259-272.
23. Balsam J, Ossandon M, Bruck HA, Lubensky I, Rasooly A. Low-cost technologies for medical diagnostics in low-resource settings. *Expert opinion on medical diagnostics*. 2013;7(3):243-255.
24. Ligler FS, Sapsford KE, Golden JP, et al. The array biosensor: portable, automated systems. *Analytical sciences : the international journal of the Japan Society for Analytical Chemistry*. 2007;23(1):5-10.
25. Ngundi MM, Qadri SA, Wallace EV, et al. Detection of Deoxynivalenol in Foods and Indoor Air Using an Array Biosensor. *Environmental Science & Technology*. 2006;40(7):2352-2356.
26. Taitt CR, Anderson GP, Ligler FS. Evanescent wave fluorescence biosensors. *Biosensors & bioelectronics*. 2005;20(12):2470-2487.
27. Moreno-Bondi MC, Taitt CR, Shriver-Lake LC, Ligler FS. Multiplexed measurement of serum antibodies using an array biosensor. *Biosensors & bioelectronics*. 2006;21(10):1880-1886.
28. Yang M, Kostov Y, Rasooly A. Carbon nanotubes based optical immunodetection of Staphylococcal Enterotoxin B (SEB) in food. *International journal of food microbiology*. 2008;127(1-2):78-83.
29. Yang M, Kostov Y, Bruck HA, Rasooly A. Carbon nanotubes with enhanced chemiluminescence immunoassay for CCD-based detection of Staphylococcal enterotoxin B in food. *Analytical chemistry*. 2008;80(22):8532-8537.
30. Yang M, Kostov Y, Bruck HA, Rasooly A. Gold nanoparticle-based enhanced chemiluminescence immunosensor for detection of Staphylococcal Enterotoxin B (SEB) in food. *International journal of food microbiology*. 2009;133(3):265-271.
31. Sapsford KE, Francis J, Sun S, Kostov Y, Rasooly A. Miniaturized 96-well ELISA chips for staphylococcal enterotoxin B detection using portable colorimetric detector. *Analytical and bioanalytical chemistry*. 2009;394(2):499-505.

32. Isikman SO, Sencan I, Mudanyali O, Bishara W, Oztoprak C, Ozcan A. Color and monochrome lensless on-chip imaging of Caenorhabditis elegans over a wide field-of-view. *Lab on a chip*. 2010;10(9):1109-1112.
33. Mudanyali O, Tseng D, Oh C, et al. Compact, light-weight and cost-effective microscope based on lensless incoherent holography for telemedicine applications. *Lab on a chip*. 2010;10(11):1417-1428.
34. Bishara W, Sikora U, Mudanyali O, et al. Holographic pixel super-resolution in portable lensless on-chip microscopy using a fiber-optic array. *Lab on a chip*. 2011;11(7):1276-1279.
35. Moon S, Keles HO, Ozcan A, et al. Integrating microfluidics and lensless imaging for point-of-care testing. *Biosensors & bioelectronics*. 2009;24(11):3208-3214.
36. Mudanyali O, Erlinger A, Seo S, Su TW, Tseng D, Ozcan A. Lensless on-chip imaging of cells provides a new tool for high-throughput cell-biology and medical diagnostics. *J Vis Exp*. 2009(34).
37. Coskun AF, Sencan I, Su TW, Ozcan A. Lensless wide-field fluorescent imaging on a chip using compressive decoding of sparse objects. *Optics express*. 2010;18(10):10510-10523.
38. Su TW, Isikman SO, Bishara W, Tseng D, Erlinger A, Ozcan A. Multi-angle lensless digital holography for depth resolved imaging on a chip. *Optics express*. 2010;18(9):9690-9711.
39. Oh C, Isikman SO, Khademhosseinieh B, Ozcan A. On-chip differential interference contrast microscopy using lensless digital holography. *Optics express*. 2010;18(5):4717-4726.
40. Coskun AF, Sencan I, Su TW, Ozcan A. Wide-field lensless fluorescent microscopy using a tapered fiber-optic faceplate on a chip. *The Analyst*. 2011.
41. Coskun AF, Nagi R, Sadeghi K, Phillips S, Ozcan A. Albumin testing in urine using a smart-phone. *Lab on a chip*. 2013;13(21):4231-4238.
42. Navruz I, Coskun AF, Wong J, et al. Smart-phone based computational microscopy using multi-frame contact imaging on a fiber-optic array. *Lab on a chip*. 2013;13(20):4015-4023.
43. Wei Q, Qi H, Luo W, et al. Fluorescent Imaging of Single Nanoparticles and Viruses on a Smart Phone. *ACS nano*. 2013;7(10):9147-9155.
44. Zhu H, Ozcan A. Wide-field Fluorescent Microscopy and Fluorescent Imaging Flow Cytometry on a Cell-phone. *Journal of Visualized Experiments*. 2013(74).
45. Zhu H, Sencan I, Wong J, et al. Cost-effective and rapid blood analysis on a cell-phone. *Lab on a chip*. 2013;13(7):1282-1288.
46. Zhu H, Yaglidere O, Su TW, Tseng D, Ozcan A. Wide-field fluorescent microscopy on a cell-phone. *Conference proceedings : Annual International Conference of the IEEE Engineering in Medicine and Biology Society IEEE Engineering in Medicine and Biology Society Annual Conference*. 2011;2011:6801-6804.
47. Breslauer DN, Maamari RN, Switz NA, Lam WA, Fletcher DA. Mobile phone based clinical microscopy for global health applications. *PloS one*. 2009;4(7):e6320.

48. Iqbal Z, Bjorklund RB. Colorimetric analysis of water and sand samples performed on a mobile phone. *Talanta*. 2011;84(4):1118-1123.
49. Tseng D, Mudanyali O, Oztoprak C, et al. Lensfree microscopy on a cellphone. *Lab on a chip*. 2010;10(14):1787-1792.
50. Zhu H, Ozcan A. Opto-fluidics based microscopy and flow cytometry on a cell phone for blood analysis. *Methods in molecular biology (Clifton, NJ)*. 2015;1256:171-190.
51. Zhu H, Yaglidere O, Su TW, Tseng D, Ozcan A. Cost-effective and compact wide-field fluorescent imaging on a cell-phone. *Lab on a chip*. 2011;11(2):315-322.
52. Balsam J, Bruck HA, Rasooly A. Webcam-based flow cytometer using wide-field imaging for low cell number detection at high throughput. *The Analyst*. 2014;139(17):4322.
53. Balsam J, Bruck HA, Rasooly A. Cell streak imaging cytometry for rare cell detection. *Biosensors and Bioelectronics*. 2015;64:154-160.
54. Ossandon MR, Phatak DS, Sorg BS, Kalpakis K. Forecasting new development of tumor areas using spatial and temporal distribution profiles of hemoglobin saturation in a mouse model. *Journal of medical imaging (Bellingham, Wash)*. 2014;1(1):014503.
55. Lowes LE, Allan AL. Recent advances in the molecular characterization of circulating tumor cells. *Cancers*. 2014;6(1):595-624.
56. Tognela A, Spring KJ, Becker T, et al. Predictive and prognostic value of circulating tumor cell detection in lung cancer: a clinician's perspective. *Critical reviews in oncology/hematology*. 2015;93(2):90-102.
57. Lu SH, Tsai WS, Chang YH, et al. Identifying cancer origin using circulating tumor cells. *Cancer biology & therapy*. 2016;17(4):430-438.
58. Obenauf AC, Massague J. Surviving at a Distance: Organ-Specific Metastasis. *Trends in cancer*. 2015;1(1):76-91.
59. de Wit S, van Dalum G, Terstappen LW. Detection of circulating tumor cells. *Scientifica (Cairo)*. 2014;2014:819362.
60. Hong B, Zu Y. Detecting circulating tumor cells: current challenges and new trends. *Theranostics*. 2013;3(6):377-394.
61. Awasthi NP, Kumari S, Neyaz A, et al. EpCAM-based Flow Cytometric Detection of Circulating Tumor Cells in Gallbladder Carcinoma Cases. *Asian Pacific journal of cancer prevention : APJCP*. 2017;18(12):3429-3437.
62. Tibbe AG, Miller MC, Terstappen LW. Statistical considerations for enumeration of circulating tumor cells. *Cytometry Part A : the journal of the International Society for Analytical Cytology*. 2007;71(3):154-162.
63. Miller MC, Doyle GV, Terstappen LW. Significance of Circulating Tumor Cells Detected by the CellSearch System in Patients with Metastatic Breast Colorectal and Prostate Cancer. *Journal of oncology*. 2010;2010:617421.
64. Man Y, Wang Q, Kemmner W. Currently Used Markers for CTC Isolation - Advantages, Limitations and Impact on Cancer Prognosis. *Journal of Clinical & Experimental Pathology*. 2011;1(1):7.

65. Truini A, Alama A, Dal Bello MG, et al. Clinical Applications of Circulating Tumor Cells in Lung Cancer Patients by CellSearch System. *Frontiers in oncology*. 2014;4:242.
66. Talasaz AH, Powell AA, Huber DE, et al. Isolating highly enriched populations of circulating epithelial cells and other rare cells from blood using a magnetic sweeper device. *Proceedings of the National Academy of Sciences of the United States of America*. 2009;106(10):3970-3975.
67. Saucedo-Zeni N, Mewes S, Niestroj R, et al. A novel method for the in vivo isolation of circulating tumor cells from peripheral blood of cancer patients using a functionalized and structured medical wire. *International journal of oncology*. 2012;41(4):1241-1250.
68. Campton DE, Ramirez AB, Nordberg JJ, et al. High-recovery visual identification and single-cell retrieval of circulating tumor cells for genomic analysis using a dual-technology platform integrated with automated immunofluorescence staining. *BMC cancer*. 2015;15:360.
69. Di Carlo D, Irimia D, Tompkins RG, Toner M. Continuous inertial focusing, ordering, and separation of particles in microchannels. *Proceedings of the National Academy of Sciences of the United States of America*. 2007;104(48):18892-18897.
70. Russom A, Gupta AK, Nagrath S, Di Carlo D, Edd JF, Toner M. Differential inertial focusing of particles in curved low-aspect-ratio microchannels. *New journal of physics*. 2009;11:75025.
71. Warkiani ME, Guan G, Luan KB, et al. Slanted spiral microfluidics for the ultra-fast, label-free isolation of circulating tumor cells. *Lab on a chip*. 2014;14(1):128-137.
72. Biosystems S. The DEPArray™ System. 2017; http://www.siliconbiosystems.com/deparray-system.
73. Fabbri F, Carloni S, Zoli W, et al. Detection and recovery of circulating colon cancer cells using a dielectrophoresis-based device: KRAS mutation status in pure CTCs. *Cancer letters*. 2013;335(1):225-231.
74. Stott SL, Hsu CH, Tsukrov DI, et al. Isolation of circulating tumor cells using a microvortex-generating herringbone-chip. *Proceedings of the National Academy of Sciences of the United States of America*. 2010;107(43):18392-18397.
75. Dharmasiri U, Balamurugan S, Adams AA, Okagbare PI, Obubuafo A, Soper SA. Highly efficient capture and enumeration of low abundance prostate cancer cells using prostate-specific membrane antigen aptamers immobilized to a polymeric microfluidic device. *Electrophoresis*. 2009;30(18):3289-3300.
76. Nagrath S, Sequist LV, Maheswaran S, et al. Isolation of rare circulating tumour cells in cancer patients by microchip technology. *Nature*. 2007;450(7173):1235-1239.
77. Xue P, Wu Y, Guo J, Kang Y. Highly efficient capture and harvest of circulating tumor cells on a microfluidic chip integrated with herringbone and micropost arrays. *Biomedical microdevices*. 2015;17(2):39.

78. Wang S, Thomas A, Lee E, Yang S, Cheng X, Liu Y. Highly efficient and selective isolation of rare tumor cells using a microfluidic chip with wavy-herringbone micro-patterned surfaces. *The Analyst*. 2016;141(7):2228-2237.
79. Shi W, Wang S, Maarouf A, et al. Magnetic particles assisted capture and release of rare circulating tumor cells using wavy-herringbone structured microfluidic devices. *Lab on a chip*. 2017;17(19):3291-3299.
80. Yang C, Zou K, Zheng L, Xiong B. Prognostic and clinicopathological significance of circulating tumor cells detected by RT-PCR in non-metastatic colorectal cancer: a meta-analysis and systematic review. *BMC cancer*. 2017;17(1):725.
81. Vaiopoulos AG, Kostakis ID, Gkioka E, et al. Detection of circulating tumor cells in colorectal and gastric cancer using a multiplex PCR assay. *Anticancer research*. 2014;34(6):3083-3092.
82. Park HS, Han HJ, Lee S, et al. Detection of Circulating Tumor Cells in Breast Cancer Patients Using Cytokeratin-19 Real-Time RT-PCR. *Yonsei medical journal*. 2017;58(1):19-26.
83. Leotsakos I, Dimopoulos P, Gkioka E, et al. Detection of circulating tumor cells in bladder cancer using multiplex PCR assays. *Anticancer research*. 2014;34(12):7415-7424.
84. De Giorgi V, Pinzani P, Salvianti F, et al. Application of a filtration- and isolation-by-size technique for the detection of circulating tumor cells in cutaneous melanoma. *The Journal of investigative dermatology*. 2010;130(10):2440-2447.
85. Hao SJ, Wan Y, Xia YQ, Zou X, Zheng SY. Size-based separation methods of circulating tumor cells. *Advanced drug delivery reviews*. 2018.
86. Sollier E, Go DE, Che J, et al. Size-selective collection of circulating tumor cells using Vortex technology. *Lab on a chip*. 2014;14(1):63-77.
87. Rostagno P, Moll JL, Bisconte JC, Caldani C. Detection of rare circulating breast cancer cells by filtration cytometry and identification by DNA content: sensitivity in an experimental model. *Anticancer research*. 1997;17(4a):2481-2485.
88. Vona G, Sabile A, Louha M, et al. Isolation by size of epithelial tumor cells : a new method for the immunomorphological and molecular characterization of circulatingtumor cells. *The American journal of pathology*. 2000;156(1):57-63.
89. Shaw Bagnall J, Byun S, Begum S, et al. Deformability of Tumor Cells versus Blood Cells. *Scientific reports*. 2015;5:18542.
90. Howell Jr PB, Golden JP, Hilliard LR, Erickson JS, Mott DR, Ligler FS. Two simple and rugged designs for creating microfluidic sheath flow. *Lab on a chip*. 2008;8(7):1097.
91. Zuba-Surma EK, Ratajczak MZ. Analytical Capabilities of the ImageStream Cytometer. *Methods in Cell Biology*: Elsevier BV; 2011:207-230.
92. Golden JP, Kim JS, Erickson JS, et al. Multi-wavelength microflow cytometer using groove-generated sheath flow. *Lab on a chip*. 2009;9(13):1942.
93. Scientific T. Fluidics of a Flow Cytometer.

https://www.thermofisher.com/us/en/home/life-science/cell-analysis/cell-analysis-learning-center/molecular-probes-school-of-fluorescence/flow-cytometry-basics/flow-cytometry-fundamentals/fluidics-flow-cytometer.html.
94. Balsam J, Bruck HA, Ossandon M, Prickril B, Rasooly A. Streak Imaging Flow Cytometer for Rare Cell Analysis. In: Rasooly A, Prickril B, eds. *Biosensors and Biodetection, Methods and Protocols Vol 1 Optical-Based Detectors*. Vol 1. Second ed. US: Springer Nature; 2017:267-286.
95. Casavant BP, Mosher R, Warrick JW, et al. A negative selection methodology using a microfluidic platform for the isolation and enumeration of circulating tumor cells. *Methods*. 2013;64(2):137-143.
96. Galanzha EI, Zharov VP. Circulating Tumor Cell Detection and Capture by Photoacoustic Flow Cytometry in Vivo and ex Vivo. *Cancers*. 2013;5(4):1691-1738.
97. Adrian RJ, Westerweel J. Low Image Density. *Particle Image Velocimetry*: Cambridge University Press; 2011:268-316.
98. Adrian RJ. Twenty years of particle image velocimetry. *Experiments in Fluids*. 2005;39(2):11.
99. Greated CA, Skyner DJ, Bruce T. Particle Image Velocimetry. Paper presented at: Coastal Engineering Proceedings1992; Venice, Italy.
100. Meijering E, Dzyubachyk O, Smal I, van Cappellen WA. Tracking in cell and developmental biology. *Seminars in cell & developmental biology*. 2009;20(8):894-902.
101. Cruz-Roa A, Caicedo JC, Gonzalez FA. Visual pattern mining in histology image collections using bag of features. *Artificial intelligence in medicine*. 2011;52(2):91-106.
102. Nunez J, Llacer J. Astronomical image segmentation by self-organizing neural networks and wavelets. *Neural networks : the official journal of the International Neural Network Society*. 2003;16(3-4):411-417.
103. Cheng D, Biao L. A Study on Geological Image Segmentation. Paper presented at: 2009 International Forum on Information Technology and Applications; 15-17 May 2009, 2009.
104. Sezgin M, Sankur Bl. Survey over image thresholding techniques and quantitative performance evaluation. *ELECTIM*. 2004;13(1):146-168.
105. Bikhet SF, Darwish AM, Tolba HA, Shaheen SI. Segmentation and classification of white blood cells. Paper presented at: 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100); 2000, 2000.
106. Reta C, Altamirano L, Gonzalez JA, et al. Segmentation and Classification of Bone Marrow Cells Images Using Contextual Information for Medical Diagnosis of Acute Leukemias. *PloS one*. 2015;10(6):e0130805.
107. Men K, Chen X, Zhang Y, et al. Deep Deconvolutional Neural Network for Target Segmentation of Nasopharyngeal Cancer in Planning Computed Tomography Images. *Frontiers in oncology*. 2017;7:315.
108. Herteleer M, Vancleef S, Nijs S, Vander Sloten J. Computed Tomography Images of the Scapula Taken With Reduced Dose Can Yield Segmented Models

of Sufficient Accuracy: A Pilot Study. *Journal of computer assisted tomography*. 2018.
109. Tian Z, Liu L, Zhang Z, Fei B. PSNet: prostate segmentation on MRI based on a convolutional neural network. *Journal of medical imaging (Bellingham, Wash)*. 2018;5(2):021208.
110. Tsujimoto M, Teramoto A, Ota S, Toyama H, Fujita H. Automated segmentation and detection of increased uptake regions in bone scintigraphy using SPECT/CT images. *Annals of nuclear medicine*. 2018.
111. Kim YJ, Lee SH, Lim KY, Kim KG. Development and Validation of Segmentation Method for Lung Cancer Volumetry on Chest CT. *Journal of digital imaging*. 2018.
112. Kumar MJ, Kumar GR, Reddy RVK. Review of Image Segmentation Techniques. *International Journal of Scientific Research Engineering & Technology*. 2014;3(6):6.
113. Hand AJ, Sun T, Barber DCH, D. R., Macneil S. Automated tracking of migrating cells in phase-contrast video microscopy sequences using image registration. *Journal of microscopy*. 2009;234(1):62-79.
114. Nattkemper TW. Automatic segmentation of digital micrographs: a survey. *Studies in health technology and informatics*. 2004;107(Pt 2):847-851.
115. Shen H, Nelson G, Kennedy S, et al. Automatic tracking of biological cells and compartments using particle filters and active contours. *Chemometrics and Intelligent Laboratory Systems*. 2006;82(1-2):276-282.
116. De S, Bhattacharyya S, Dutta P. Image Segmentation: A Review. *Hybrid Soft Computing for Multilevel Image and Data*: Springer International Publishing; 2016:29-40.
117. Anjna E, Kaur ER. Review of Image Segmentation Technique. *International Journal of Advanced Research in Computer Science*. 2017;8(4):4.
118. Kumbhar PGH, Sushilkumar N. A Review of Image Thresholding Techniques. *International Journal of Advanced Research in Computer Science and Software Engineering*. 2015;5(6):4.
119. Zhou F, Jia Z, Yang J, Kasabov N. Method of Improved Fuzzy Contrast Combined Adaptive Threshold in NSCT for Medical Image Enhancement. *BioMed research international*. 2017;2017:3969152.
120. Wang R, Zhou Y, Zhao C, Wu H. A hybrid flower pollination algorithm based modified randomized location for multi-threshold medical image segmentation. *Bio-medical materials and engineering*. 2015;26 Suppl 1:S1345-1351.
121. Liu Y, Liu J, Tian L, Ma L. Hybrid Artificial Root Foraging Optimizer Based Multilevel Threshold for Image Segmentation. *Computational intelligence and neuroscience*. 2016;2016:1476838.
122. Liu P, Chen Q, Ma J. Design of [2]rotaxane through image threshold segmentation of electrostatic potential image. *Journal of computational chemistry*. 2016;37(24):2228-2241.
123. Kang D, Hua H, Peng N, Zhao J, Wang Z. Improving Image Quality of Coronary Computed Tomography Angiography Using Patient Weight and Height-Dependent Scan Trigger Threshold. *Academic radiology*. 2017;24(4):462-469.

124. Issac A, Partha Sarathi M, Dutta MK. An adaptive threshold based image processing technique for improved glaucoma detection and classification. *Computer methods and programs in biomedicine*. 2015;122(2):229-244.
125. Chunming Z, Yongchun X, Da L, Li W. Fast Threshold Image Segmentation Based on 2D Fuzzy Fisher and Random Local Optimized QPSO. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*. 2017;26(3):1355-1362.
126. Seo HU, Wei Q, Kwon SG, Sohng KI. Medical image watermarking using bit threshold map based on just noticeable distortion in discrete cosine transform. *Technology and health care : official journal of the European Society for Engineering and Medicine*. 2017;25(S1):367-375.
127. Meijering E, Smal I, Dzyubachyk O, Olivo-Marin J-C. Time-Lapse Imaging. *Microscope Image Processing*. USA: Elsevier Academic Press; 2008:401-440.
128. Meijering E, Dzyubachyk O, Smal I. Chapter nine - Methods for Cell and Particle Tracking. In: conn PM, ed. *Methods in Enzymology*. Vol 504: Academic Press; 2012:183-200.
129. Er. Nirpjeet kaur ERk. A Review on Various Methods of Image Thresholding *International Journal on Computer Science and Engineering (IJCSE)*. 2011;3(10):3.
130. Rafael C. Golzalez REW. Image Segmentation. *Digital Imaging Processing*. USA: Pearson Prentice Hall; 2008:689-787.
131. Senthilkumaran N VS. Image Segmentation by Using Thresholding Techniques for Medical Images. *Computer Science & Engineering: An International Journal (CSEIJ)*. 2016;6(1):13.
132. Otsu N. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*. 1979;9(1):62-66.
133. Greensted A. Otsu Thresholding. 2010; http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html.
134. Cai H, Yang Z, Cao X, Xia W, Xu X. A New Iterative Triclass Thresholding Technique in Image Segmentation. *IEEE Transactions on Image Processing*.23(3):9.
135. Pal NR, Pal SK. A review on image segmentation techniques. *Pattern Recognition*. 1993;26(9):1277-1294.
136. Bhadauria H, Singh A, Kumar A. Comparison between various edge detection methods on satellite image. *International Journal of Emerging Technology and Advanced Engineering*. 2013;3(6):324-328.
137. Zhang M, Li X, Yang Z, Yang Y. A novel zero-crossing edge detection method based on multi-scale space theory. Paper presented at: Signal Processing (ICSP), 2010 IEEE 10th International Conference on2010.
138. Shrivakshan G, Chandrasekar C. A comparison of various edge detection techniques used in image processing. *IJCSI International Journal of Computer Science Issues*. 2012;9(5):272-276.
139. Canny J. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1986;PAMI-8(6):679-698.
140. Sornam M, Kavitha MS, Nivetha M. Hysteresis thresholding based edge detectors for inscriptional image enhancement. Paper presented at: 2016 IEEE

International Conference on Computational Intelligence and Computing Research (ICCIC); 15-17 Dec. 2016, 2016.
141. Perez SVF, Lara L, Gonzalez M. A Threshold with Hysteresis. *The Insight Journal*. 2011.
142. Wahlby C, Sintorn IM, Erlandsson F, Borgefors G, Bengtsson E. Combining intensity, edge and shape information for 2D and 3D segmentation of cell nuclei in tissue sections. *Journal of microscopy*. 2004;215(Pt 1):67-76.
143. Zhou X, Li F, Yan J, Wong STC. A Novel Cell Segmentation Method and Cell Phase Identification Using Markov Model. *IEEE Transactions on Information Technology in Biomedicine*. 2009;13(2):6.
144. Vincent L, Soille P. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1991;13(6):11.
145. Zimmer C, Labruyere E, Meas-Yedid V, Guillen N, Olivo-Marin JC. Segmentation and tracking of migrating cells in videomicroscopy with parametric active contours: a tool for cell-based drug testing. *IEEE transactions on medical imaging*. 2002;21(10):1212-1221.
146. Debeir O, Van Ham P, Kiss R, Decaestecker C. Tracking of migrating cells under phase-contrast video microscopy with combined mean-shift processes. *IEEE transactions on medical imaging*. 2005;24(6):697-711.
147. O'Hara S, Draper BA. Introduction to the Bag of Features Paradigm for Image Classification and Retrieval. *CoRR abs/11013354*. 2011.
148. Jianchao Y, Kai Y, Yihong G, Huang T. Linear spatial pyramid matching using sparse coding for image classification. Paper presented at: 2009 IEEE Conference on Computer Vision and Pattern Recognition; 20-25 June 2009, 2009.
149. Meijering E, Dzyubachyk O, Smal I. Methods for Cell and Particle Tracking. *Imaging and Spectroscopic Analysis of Living Cells - Optical and Spectroscopic Techniques*: Elsevier BV; 2012:183-200.
150. Huth J, Buchholz M, Kraus JM, et al. TimeLapseAnalyzer: Multi-target analysis for live-cell imaging and time-lapse microscopy. *Computer methods and programs in biomedicine*. 2011;104(2):227-234.
151. Schneider CA, Rasband WS, Eliceiri KW. NIH Image to ImageJ: 25 years of image analysis. *Nature methods*. 2012;9(7):671-675.
152. Cordelieres FP, Petit V, Kumasaka M, et al. Automated cell tracking and analysis in phase-contrast videos (iTrack4U): development of Java software based on combined mean-shift processes. *PloS one*. 2013;8(11):e81266.
153. Stuurman N. MTrack2. https://valelab4.ucsf.edu/~nstuurman/IJplugins/MTrack2.html.
154. Sacan A, Ferhatosmanoglu H, Coskun H. CellTrack: an open-source software for cell tracking and motility analysis. *Bioinformatics*. 2008;24(14):1647-1649.
155. Piccinini F, Kiss A, Horvath P. CellTracker (not only) for dummies. *Bioinformatics*. 2016;32(6):3.
156. Klopfenstein DR, Vale RD. The lipid binding pleckstrin homology domain in UNC-104 kinesin is necessary for synaptic vesicle transport in Caenorhabditis elegans. *Molecular biology of the cell*. 2004;15(8):3729-3739.

157. Kuhn J. MultiTracker. 2001; https://imagej.nih.gov/ij/plugins/multitracker.html.
158. Rasband W. Object Tracker. 2000; https://imagej.nih.gov/ij/plugins/tracker.html.
159. Ossandon M, Balsam J, Bruck HA, Rasooly A, Kalpakis K. Evaluation of a Methodology for Automated Cell Counting for Streak Mode Imaging Flow Cytometry. *Journal of Analytical & Bioanalytical Techniques*. 2017.
160. MathWorks. MATLAB Image Coordinate Systems. https://www.mathworks.com/help/images/image-coordinate-systems.html#brcu_cr.
161. Ossandon M, Balsam J, Bruck HA, Kalpakis K, Rasooly A. A computational streak mode cytometry biosensor for rare cell analysis. *The Analyst*. 2017.
162. Sørensen TJ, Thyrhaug E, Szabelski M, et al. Azadioxatriangulenium (ADOTA(+)): A long fluorescence lifetime fluorophore for large biomolecule binding assay. *Methods and applications in fluorescence*. 2013;1(2):025001.
163. Rich RM, Stankowska DL, Maliwal BP, et al. Elimination of autofluorescence background from fluorescence tissue images by use of time-gated detection and the AzaDiOxaTriAngulenium (ADOTA) fluorophore. *Analytical and bioanalytical chemistry*. 2013;405(6):2065-2075.
164. Tang J, Li X, Price MA, et al. CD4:CD8 lymphocyte ratio as a quantitative measure of immunologic health in HIV-1 infection: findings from an African cohort with prospective data. *Frontiers in Microbiology*. 2015;6:670.
165. Zhang YS, Ribas J, Nadhman A, et al. A Cost-Effective Fluorescence Mini-Microscope with Adjustable Magnifications for Biomedical Applications. *Lab on a chip*. 2015;15(18):3661-3669.
166. Zhang YS, Santiago GT, Alvarez MM, Schiff SJ, Boyden ES, Khademhosseini A. Expansion Mini-Microscopy: An Enabling Alternative in Point-of-Care Diagnostics. *Current opinion in biomedical engineering*. 2017;1:45-53.
167. Proserpio V, Lönnberg T. Single-cell technologies are revolutionizing the approach to rare cells. *Immunology and Cell Biology*. 2016;94(3):225-229.
168. Nguyen A, Khoo WH, Moran I, Croucher PI, Phan TG. Single Cell RNA Sequencing of Rare Immune Cell Populations. *Frontiers in immunology*. 2018;9:1553.
169. (WHO) WHO. The top 10 causes of death. 2018; http://www.who.int/en/news-room/fact-sheets/detail/the-top-10-causes-of-death. Accessed 8/27/2018, 2018.
170. Baharom F, Rankin G, Blomberg A, Smed-Sörensen A. Human Lung Mononuclear Phagocytes in Health and Disease. *Frontiers in immunology*. 2017;8:499.
171. (WHO) WHO. Antibiotic resistance. 2018; http://www.who.int/news-room/fact-sheets/detail/antibiotic-resistance. Accessed 8/16/2018.
172. Ventola CL. The Antibiotic Resistance Crisis: Part 1: Causes and Threats. *Pharmacy and Therapeutics*. 2015;40(4):277-283.
173. Bebell LM, Muiru AN. Antibiotic use and emerging resistance—how can resource-limited countries turn the tide? *Global heart*. 2014;9(3):347-358.

174. Baltekin Ö, Boucharin A, Tano E, Andersson DI, Elf J. Antibiotic susceptibility testing in less than 30 min using direct single-cell imaging. *Proceedings of the National Academy of Sciences of the United States of America*. 2017;114(34):9170-9175.
175. Jorgensen JH, Ferraro MJ. Antimicrobial susceptibility testing: a review of general principles and contemporary practices. *Clinical infectious diseases : an official publication of the Infectious Diseases Society of America*. 2009;49(11):1749-1755.
176. Coorevits L, Boelens J, Claeys G. Direct susceptibility testing by disk diffusion on clinical samples: a rapid and accurate tool for antibiotic stewardship. *European Journal of Clinical Microbiology & Infectious Diseases*. 2015;34(6):1207-1212.
177. Erdem SS, Khan S, Palanisami A, Hasan T. Rapid, low-cost fluorescent assay of beta-lactamase-derived antibiotic resistance and related antibiotic susceptibility. *Journal of biomedical optics*. 2014;19(10):105007.
178. Bain R, Cronk R, Wright J, Yang H, Slaymaker T, Bartram J. Fecal contamination of drinking-water in low- and middle-income countries: a systematic review and meta-analysis. *PLoS medicine*. 2014;11(5):e1001644.
179. Orma B. Water Testing Bacteria, Coliform, Nuisance Bacteria, Viruses, and Pathogens in Drinking Water. 2014; https://www.water-research.net/index.php/bacteria. Accessed 058/22/2018, 2018.