

Reverse Engineering of the Actin Cortex with Speckle Microscopy and Smart Capture

Nargess Khalilgharibi, CoMPLEX

Supervisors: Dr Guillaume Charras & Dr Gabriel Brostow

Total Word count: 4824

Main Body: 5169

(Excluding abstract, figure captions, displayed equations, acknowledgement, references and appendix)

May 2, 2012

In 1939, W H Lewis was one of the pioneers to suggest the presence of a "superficial plasmagel" at the membrane of the cell, which contributed to its locomotion [3]. Although the light microscopy at that time was unable to identify this layer, it is now well known that this layer is the actin cortex, a meshwork of actin filaments, responsible for determining the shape and movement of the cell [1].

Although many of the actin binding proteins and their functions have been determined, much still remains to be discovered. Images obtained from Fluorescent Speckle Microscopy (FSM) can provide a lot of information about the assembly of the cell cortex, as well as the actin dynamics. However, it is not possible to extract these information from FSM images without an accurate tracking algorithm.

In the following essay, a new tracking algorithm will be developed and used to track actin speckles.

Contents

1	Introduction	1
2	Actin Cortex: The Underlying Biology	1
2.1	From Monomeric G-Actin to 3D Actin Cortex	1
2.2	Blebbing	2
3	Fluorescent Speckle Microscopy	3
4	Smart Capture: Application to Speckle Tracking	4
4.1	Particle Filter	4
4.2	Tracking the Speckle Trajectories using the Particle Filter	5
5	Results	7
5.1	Tracking	7
5.2	Measuring the Intensity Distribution of A Single Speckle	10
6	Discussion and Suggestions for Further Research	11
7	Acknowledgements	12
A	Appendix	13
A.1	The Speckle Tracking Algorithm	13
A.1.1	Main Function of the Algorithm	14
A.2	The Algorithm for Determining the Sigmoid Function	20
A.3	The Algorithm for Determining the PSF Function of a Single Speckle	21

1 Introduction

The cytoskeleton, the “skeleton” which protects the shape of the eukaryotic cell and contributes to its structure and mechanical functions [1] is formed of three types of protein filaments. One type of these filaments, i.e. actin filaments, are helical threads of globular actin molecules [1]. The actin cortex is a 50-nm-2- μ m-thick [6] cytoskeletal layer beneath the plasma membrane with a high concentration of actin filaments, myosin and actin binding proteins [1]. It is responsible for determining the “shape of the cell surface” [1] and its morphogenesis [6], necessary for the cell motility and movement [2, 6] and involved in cytokinesis [6]. Despite its functionality, the mechanisms which lead to organisation and assembly of the actin cortex [6], as well as the exact role of different actin binding proteins in its formation [3, 4] are not well understood.

Fluorescence microscopy has revealed huge amounts of information about protein localization [8, 26] and the dynamics of macromolecular assemblies [8]. However, high background fluorescence and uniform labelling of structures limits its ability to study protein dynamics and prevents the detection of movement [8, 26]. Different methods such as three dimensional confocal microscopy [26], Fluorescence Recovery After Photobleaching (FRAP) and Photoactivation of Fluorescence (PAF) [8, 26] have been utilised to partially solve these problems. However, these methods have their own limitations [24]. In addition to providing the same information as previous methods, Fluorescent Speckle Microscopy (FSM) can be used to study assembly dynamics and protein movement and turnover within assemblies at high spatial and temporal resolution [8, 26]. In particular, FSM has been used to study the actin turnover [25], as well as the role of different proteins in actin nucleation [13, 17, 23].

An important step in live-cell imaging is to track the trajectories of particles. In fact, the studies of subcellular dynamics (e.g. actin dynamics) would only be accurate if the motion of particles could be identified with a good precision. High concentration of particles, their heterogenous motion, temporary disappearance, merging and splitting makes this an even more challenging step [14]. This indicates the need for tracking algorithms, which could provide good estimations of the particles’ trajectories.

In the following essay, I will briefly explain the structure of the actin cortex and give a description of blebs, one of the protrusions of the cell membrane (Section 2). I will proceed with introducing the fluorescent speckle microscopy, which was used in this study to image blebbing cells and actin molecules in the cell (Section 3). The essay will continue with a section on conventional tracking algorithms and particle filtering technique (Section 4 and 4.1). I will then introduce a tracking algorithm based upon the particle filter (Section 4.2). The results obtained from using this algorithm to track the actin speckles are presented in Section 5. The essay will end with a discussion on possible issues of this algorithm and suggestions for future work (Section 6).

2 Actin Cortex: The Underlying Biology

2.1 From Monomeric G-Actin to 3D Actin Cortex

Actin can be either found as monomeric globular actin (G-actin) or polymeric filamentous actin (F-actin) [24]. In the cell, there is roughly an equal number of G-actin and F-actin [24]. A few actin subunits can bind together to form short oligomers, but these structures are unstable [1]. In order to form long stable filaments, an initial stabilized nucleus, consisting of 4 actin monomers is needed [24]. This step, known as nucleation, is a rate limiting and energetically costly step [1]. Actin nucleation can be regulated by several actin nucleating factors (e.g. Arp2/3 complex, formins) [1, 24]. This can be advantageous for the cell, enabling it to control its shape and movements by forming new actin filaments in specific locations where actin nucleating factors are present [1]. The second step in formation of actin filaments is elongation, in which actin monomers are added to the ends of nucleated filaments. When a steady state is reached, the rate of actin addition to the filament equals the rate of actin dissociation [1].

By interacting with various actin binding proteins, actin filaments can form different structures. Two groups of actin binding proteins, i.e. actin bundling and cross-linking proteins help bind actin filaments

together in such a way to form the three dimensional gel-like meshwork of actin cortex [2, 27]. This meshwork of actin filaments is tangent to the cell surface [6] and helps the cell to maintain its shape and resist deformation [3]. Remodelling of this actin network results in changes in the shape of the cell and its locomotion [24].

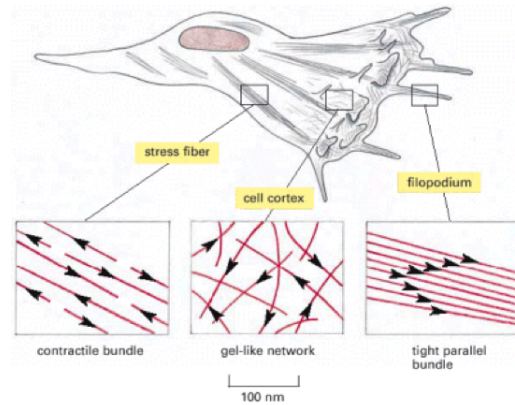


Figure 1: Different arrangements of the actin filaments taken from [1].

Protrusions of the cell membrane contribute to its locomotion. One type of these protrusions is lamellipodia, a dynamic structure of actin filaments, formed on the leading edge of motile cells and used for crawling on a surface [1, 2]. For a long time, lamellipodial motility has been known as the major migration mechanism in the cells [5]. However, recent studies have drawn attention to blebbing, another motility mechanism which is an alternative to lamellipodial protrusion, especially in three dimensional environment [4, 5]. Unlike lamellipodia, which is formed by actin polymerization, growth of blebs is pressure-driven [5, 24]. In addition to being an essential movement mechanism for some cells, blebbing can be considered as "a window" into the assembly of actin cortex, because blebs are one of the few natural occasions where "cortex-free membrane" could be found [6].

2.2 Blebbing

The formation of blebs can be divided into three major stages (Figure 2). In the first stage, called bleb nucleation (initiation), the actin cortex is decoupled from the membrane by the actomyosin contraction mechanism¹ [4]. In fact, actomyosin contraction increases the intracellular pressure, either leading to delimitation of cortex from the membrane or rupture of the cortex.

In the second stage, bleb expansion, the cytosol enters the bleb, resulting in its rapid volume increase. Following the increase in the bleb's volume, its surface area should also grow. This can be done through several mechanisms. For example, it has been observed that the base of the bleb grows over time, indicating that increase in surface area could be due to delimitation of the cell membrane from the actin cortex [4]. Also, the membrane of a cell is a folded and wrinkled surface, storing a large excess of membrane. Unfolding of these membrane wrinkles leads to an increase in the surface area [4]. Furthermore, the increase in the surface area could be a result of the flow of lipids into the bleb through the bleb neck [4, 5]. It is still not known how much each of these mechanisms contribute to the growth in bleb's surface area [4]. Actin cortex has not been observed in growing blebs, suggesting that blebs may be devoid of actin cortex [5]. Although this could be due to limitations of optical resolution [5], confocal microscopy images of blebbing M2 cells expressing GFP-actin have confirmed low concentration of actin in growing blebs [6].

The last stage in the life of the bleb is called bleb retraction. During this stage, the actin cortex is once

¹The actomyosin contraction is the contraction of the cell cortex due to contractile forces exerted on it by myosin motors [6, 19]

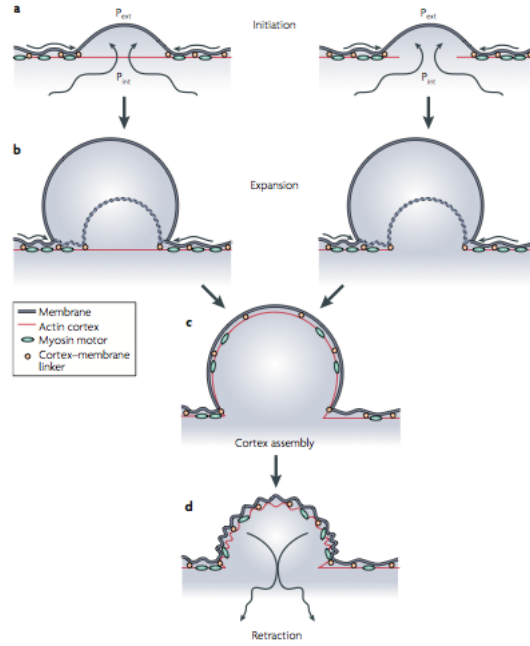


Figure 2: The life cycle of a bleb taken from [5].

again assembled beneath the membrane, forcing the bleb to retract. It is not clear how actin starts to nucleate beneath the membrane, because two well-known actin nucleating factors (i.e. Arp2/3 and mDia1) did not localise to the membrane [6]. It has been suggested that a different formin (mDia2) may contribute to cortical actin nucleation [10]. It is also possible that the remainder of filamentous actin attached to the membrane regulate actin nucleation [4]. After the reassembly of the actin cortex and recruitment of actin-bundling and contractile proteins, myosin II is recruited, powering bleb retraction [4, 5, 6]. It should be noted that in motile cells, the retraction stage does not always happen. In fact, in these cells, polarised blebbing at the leading edge of the cell [4], followed by contraction at the rear [5] leads to motion.

3 Fluorescent Speckle Microscopy

Fluorescent Speckle Microscopy (FSM) is a microscopy method combining low concentration of fluorescent probes, fluorescent light microscopy and cooled Charged Coupled Device (CCD) camera to provide high resolution images of molecular dynamics *in vivo* and *in vitro* [8, 26]. The core principle in FSM is utilising a small fraction of fluorescently tagged probes compared to the amount of unlabelled molecules. This reduces the level of background fluorescence [26].

A speckle is a "diffraction-limited region" in the image, where the fluorescent intensity is peaked due to higher concentration of fluorophores compared to the background regions [8]. The occurrence of speckles in the image is due to the convolution of the fluorophore distribution with the Point Spread Function (PSF) of the imaging device [8]. It has been shown that only immobilised fluorophores, i.e. those fluorophores which stay within the PSF volume during the exposure time of the imaging device (~ 0.1 to 2 s), will give rise to speckles. Free, diffusible fluorescent molecules will give rise to a uniformly distributed background light across the pixels they visit during the exposure time [8, 24]. Changes in the speckle distribution implies movement of the tagged molecules, while changes in intensity indicates their assembly [26].

In the actin cortex, fluorescently tagged actins associate with the three dimensional meshwork of actin filaments. Due to being incorporated into the cortex, these actins are immobilised and thus, give rise to speckles [8]. The diffusible labelled actin monomers, dissociated from the actin network, will give rise

to low intensity evenly distributed signals, which will be blurred over several pixels and will disappear in the background [8, 25]. Since the mesh size of the actin network is usually below the resolution limit of conventional microscopes, signals from several filaments that fall into one PSF volume give rise to a speckle [8].

4 Smart Capture: Application to Speckle Tracking

The speckle images are rich sources of information about the subcellular dynamics, as well as the movement of the speckle platform² [8]. In order to derive these information, the speckles' displacements should be tracked over time. In early studies, the speckles were tracked manually (e.g. [17, 23, 25]). Although this was possible for a few speckles, in images with high concentration of speckles, hand tracking became complicated and erroneous [8]. Thus, algorithms which could automatically track the speckles were required.

Attempts to develop automated tracking algorithms for studying molecular compounds and single molecule dynamics goes back to early 1980s [15]. The structure of many of these algorithms are the same [15]. In fact, many of the developed speckle tracking algorithms consist of two major parts: detection and tracking. Detection step aims to evaluate the positions of the speckles in all frames. Detection can be done through different approaches but one of the most effective approaches is to fit a predefined mathematical model of the speckle intensity distribution to the data [15]. The aim of the tracking step is to link the particles between consecutive frames, in order to extract the trajectories of speckles.

Many of the particle tracking algorithms follow a probabilistic approach, which is based upon the Bayesian model [16]. The goal of the Bayesian model is to infer the state of an object based on all the available observations. This can be done by estimating a "time evolving filtering distribution" [21] that can provide the state of an object at a certain time, given all the measurements up to that time [12, 21]. Different filtering techniques have been developed to implement the Bayesian approach (e.g. Kalman filter, extended Kalman filter, unscented Kalman filter, particle filter).

4.1 Particle Filter

The particle filter is a filtering technique that could be particularly applied to multi-modal problems [12, 20]. In the particle filtering method, a set of particles with associated weights represent the possible states of an object. By propagating these particles through time, the filtering distribution could be approximated at subsequent times [20, 21].

One of the methods to implement the particle filter is "condensation algorithm" [20]. Consider the states of the object to be represented by $\{w_t\}_{t=1}^T$ and the observations by $\{x_t\}_{t=1}^T$. The filtering distribution can be estimated as:

$$P(w_{t-1}|x_{1...t-1}) \approx \sum_{j=1}^J a_{t-1}^j \delta(w_{t-1} - w_{t-1}^j) \quad (1)$$

where $\{w_t^j, a_t^j\}_{j=1}^J$ are J particles and their associated weights, which sum to one. The process of calculating $P(w_t|x_{1...t})$ consists of two steps. The goal of the first step, the time evolution step, is to create a set of J unweighted particles by resampling the weighted particles from the previous time step. Resampling can be done by choosing J particles from the set of original weighted particles, according to their weight. Since the weight of the particles represent their probability (or our confidence in them), the highest weighted particles are more likely to be chosen for subsequent time steps; thus, contributing more to the final set [20]. After choosing the particles, their state will be updated by the motion model

²The speckle platform is a scaffold with which the fluorophores are associated, giving rise to speckles.

(also known as the dynamic model). The new particles will be the temporal updates of the chosen particles. In the second step, the measurement incorporation step, the new set of particles will be weighted based on their accordance with observations. In other words, the particles will be passed through the measurement model (also known as the appearance model) and weighted based upon how well they agree with the measurements. Then, the weights are normalised so that they sum to one [20]. Finally, the state of the particle with highest weight will be assigned to w_t , the new state of the object. Figure 3 illustrates the different steps of the condensation algorithm.

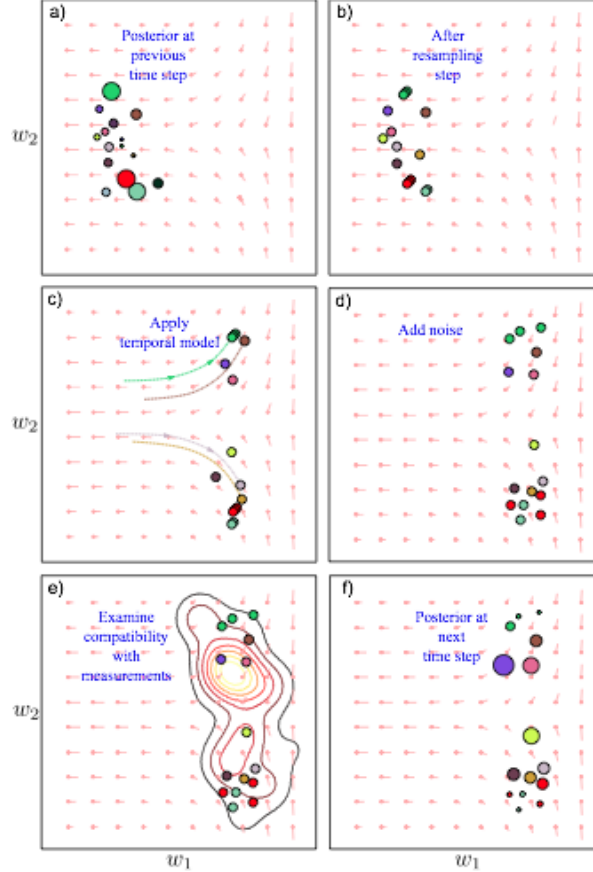


Figure 3: Different steps of the condensation algorithm. a) Set of weight particles from the previous step b) A new set of unweighted particles is generated by resampling the particles from the previous step according to their weights c) The state of the particles is updated by the motion model. d) Noise is added to the states of the particles. e) Weight is assigned to each particle by the appearance model. f) A new set of weighted particles has been generated. Figure taken from [20]

4.2 Tracking the Speckle Trajectories using the Particle Filter

The algorithm which was previously used to track the actin speckles, was developed by Jaqaman et al [14]. Similar to other conventional tracking algorithms, this algorithm consisted of the detection and tracking steps. In the tracking step, the Kalman filter was used to find the correspondence between the speckles in subsequent frames.

Tracking the actin speckles in images obtained from FSM can be considered as a multi-modal problem, because these images usually contain a huge number of speckles. The Kalman filter is suitable for tracking particles in systems that have linear dynamics [16] and whose measurement noise has a normal distribution [20]. Thus, it is not suitable for problems where "the true probability distribution over the state is multi-modal" [20]. Since the particle filter can cope with the complexities of multi-modal track-

ing [20], it seemed suitable for tracking the actin speckles.

In order to investigate whether the particle filter was a better alternative than the Kalman filter, an algorithm based upon particle filtering technique was provided. This algorithm was developed by Dr Gabriel Brostow and Cristina Garcia-Cifuentes to track objects in colour images. A brief description of the algorithm is as follows.

First, the movies and the initial positions of the objects to be tracked are given as input to the algorithm. The particle filtering part includes initiation, temporal propagation and weighting of 50 particles, using the condensation algorithm. Six motion models are included in the algorithm: Brownian, constant velocity, upward, downward, leftward and rightward motions. The appearance model uses normalised cross correlation (NCC) ³ to assign weights to the particles. In fact, the cross correlation between a 15 by 15 patch around each particle and a patch of the same size around the initial position of the object is calculated and used as a measure of similarity.

In order to enable it to read grayscale FSM images, the algorithm was modified. This algorithm was then used to track speckles in five movies. Three motion models, i.e. Brownian, constant velocity and downward motion, were used for this purpose. The algorithm was programmed such that it displayed the true positions of the speckles (given by the ground truth data), their probable positions (given by the particles) and their predicted positions in each frame. Thus, it was possible to compare the predicted tracks with the ground truth, while the movies were displayed. However, the results of the tracking were not satisfying. The algorithm lost track of the speckles after two or three frames. Since for each speckle, some of the particles were better estimates of its position than the chosen one, it was concluded that the appearance model failed to choose the best particle. This could be due to choosing the local intensity distribution around the initial position of the speckle as template for NCC. As shown in figure 4, the signal of the speckle decays during time [25]. Due to this temporal change in speckle's intensity (and sometimes its shape), the particle whose intensity distribution is more similar to that around the initial position of the speckle may not necessarily be the best choice for its subsequent position.

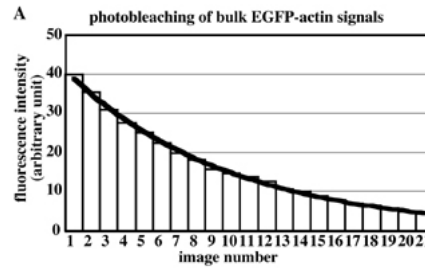


Figure 4: Exponential decay of fluorescence in a cell expressing EGFP-actin, taken from [25].

As a first step to resolve this problem, the algorithm was modified such that at each step, instead of comparing the intensity distribution around each particle with the first frame, its intensity would be compared with that of the highest weight particle in the previous step. It was expected that this would improve the tracking, because there is not a huge change in the shape and intensity of the speckles between two subsequent frames and thus, the similarity between two points may suggest that they are in fact showing one speckle in two different time steps. However, when used to track the speckles, the algorithm did not give contenting results.

Thus, it was necessary to develop another appearance model. The idea was that for small actin speckles,

³The normalised cross correlation between an $M \times N$ template image patch I and a displaced patch J can be calculated as [22]:

$$NCC = \frac{\sum_{i=1}^M \sum_{j=1}^N [I_{ij} - \bar{I}][J_{ij} - \bar{J}]}{\sqrt{\sum_{i=1}^M \sum_{j=1}^N [I_{ij} - \bar{I}]^2 [J_{ij} - \bar{J}]^2}}$$

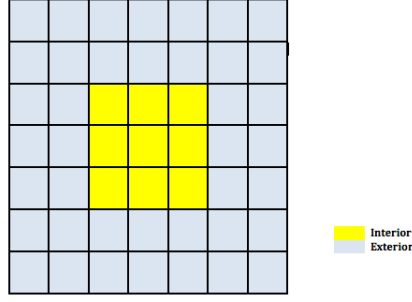


Figure 5: The pixels around the position of each speckle were divided into interior and exterior pixels. Then the difference between their average intensities was calculated. The central cell shows the position of the speckle.

there would be a huge difference between the average intensities of a small patch close to the speckle and a bigger patch around it, while for non-speckling points, as well as huge speckling vesicles, there would not be such intensity difference. Hence, this difference in intensity could be used as a measure for deciding whether a point was an actin speckle or not. In order to find a function that could make this distinction, a machine learning approach was taken. A set of 136 speckling and 143 non-speckling points were provided from six FSM movies of the actin cortex. These points were used as training data to develop a function which could distinguish between speckling and non-speckling points. As shown in figure 5, the pixels in a 3×3 patch around the position of each point were called interior pixels and the pixels in a 7×7 patch, excluding those in the interior, were called exterior pixels. The average intensities of the interior and exterior pixels were calculated (Figure 6). For speckling points, the difference between the average intensities of the interior and exterior pixels was large (averaged 0.096^4), while for non-speckling points, this difference was small (averaged -0.005^4). In order to convert these differences into probabilities, a sigmoid function was used. The sigmoid function is a logistic function which can be shown as:

$$f(x) = \frac{1}{1 + e^{-a(x-c)}} \quad (2)$$

where a is the slope parameter of the function and c is the bias [9]. The slope parameter a was set to 1. In order to find c , the speckling points were given a probability 1 and non-speckling points were given a probability 0. These probabilities were plotted versus the difference between interior and exterior intensities, and different c s were examined to find the best sigmoidal fit to the plot. Finally, the sigmoid function which could give the probability of a point to be a speckle, based on the difference between its interior and exterior intensities was found as:

$$P(\Delta) = \frac{1}{1 + e^{-(\Delta-2108)}} \quad (3)$$

where Δ is the difference between the average intensities of the interior and exterior pixels. This function was then implemented in the tracking algorithm, to assign weights to the particles. Finally, the modified tracking algorithm was used to track the actin speckles in five FSM movies. It should be mentioned that in order to improve the tracking results, the number of particles was increased from 50 to 100.

5 Results

5.1 Tracking

Sixteen speckles from five movies, for which the ground truth data was available, were chosen for tracking. Seven of these speckles displayed actin near the periphery of non-blebbing cells. The other nine were of actin speckles inside blebs. Different motion models, i.e. Brownian, constant velocity and downward motion were examined. For six of the seven speckles, the Brownian motion was the best motion

⁴Both values are normalised by dividing the difference value by 65535, the maximum intensity for 16-bit grayscale images.

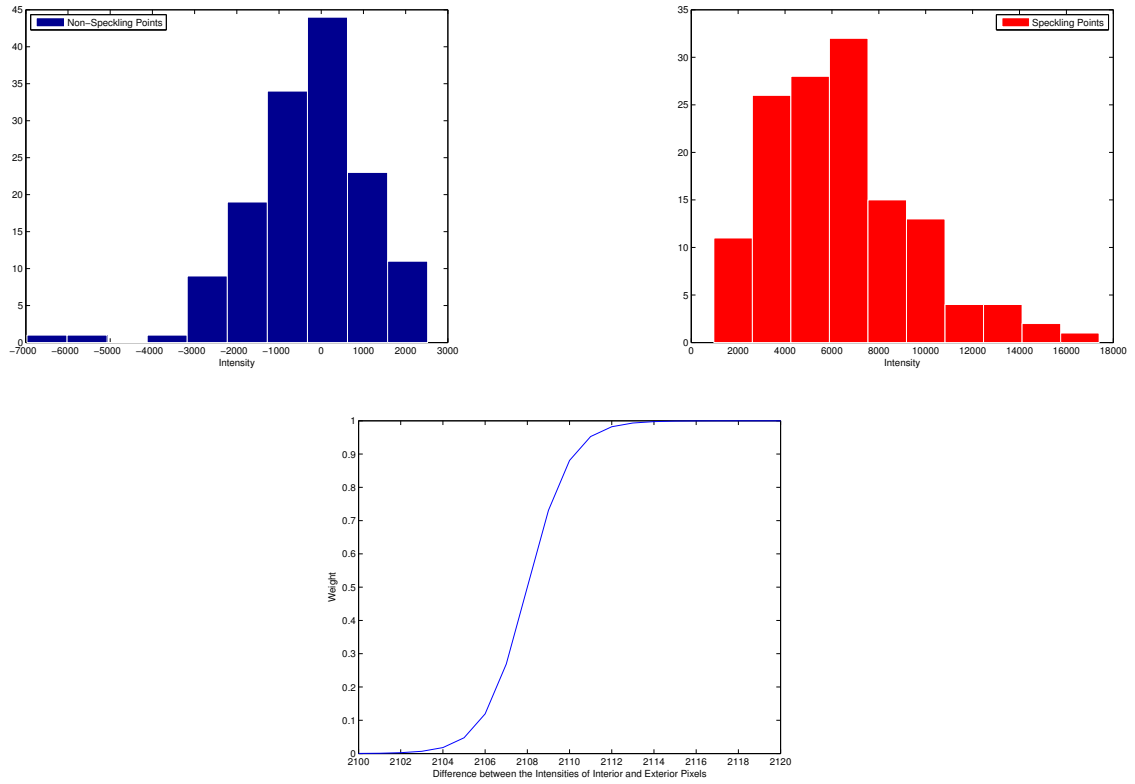


Figure 6: **(Top)** The histogram of the difference between the intensities of interior and exterior pixels for (Left) non Speckling points (Right) Speckling Points. **(Bottom)** The sigmoid function which was used to assign weights to particles

model. In fact, for this motion model, the Root Mean Square Error (RMSE)⁵ between the tracked and actual positions of the speckles was 1.23 pixels⁶ on average. However, for the nine speckles from blebbing cells, neither the Brownian motion, nor the constant velocity were able to track the speckles, having RMSE of 2.42 and 2.10 pixels, respectively. For these speckles, tracking was improved by using the downward motion model, which gave an RMSE of 1.50 pixels. This seemed reasonable, because the blebs were contracting and the actin speckles, which were attached to the cortex of the blebs, were moving downward with it, towards the cell body.

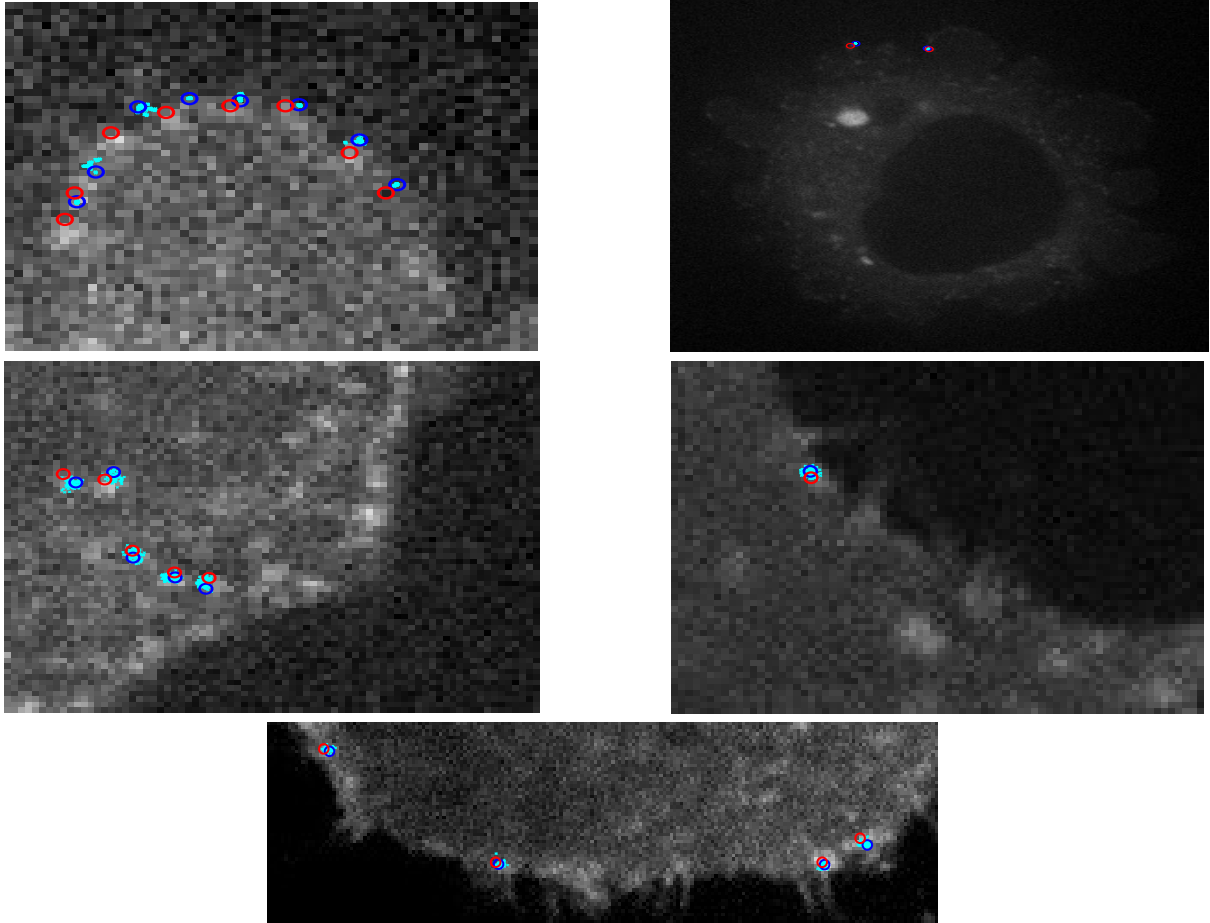


Figure 7: Images of tracked speckles. The red circles show the positions of the speckles given by the ground truth data, while the blue circles display their predicted positions. The light blue dots show the positions of particles. **(Top)** Blebbing Cells **(Middle and Bottom)** Non-blebbing Cells

To evaluate whether speckle tracking had improved by use of particle filtering technique, comparisons were made with the results obtained from the algorithm developed by Jaqaman et al [14]. Since this algorithm detected the speckles automatically, it was unable to detect and track the speckles in one of the blebbing cells, for which the ground truth data was provided. Thus, eight speckles from the remaining four movies were used for making the comparisons. Overall, the algorithm which was developed in this research was able to give better estimates of the positions of the speckles. In fact, the average RMSE of this algorithm was 1.15 pixels, while the average RMSE of Jaqaman et al's algorithm was 3.33 pixels. To be more specific, for five speckles, the predictions of the algorithm developed in this research were more accurate than that of Jaqaman et al. For two speckles, the RMSE of the two algorithms were

⁵RMSE was calculated as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (x_{tracking,i} - x_{groundtruth,i})^2}{n}}$$

⁶For the imaging devices used in this research, 1 pixel equals 133 nanometers.

roughly the same (0.82 and 0.78 pixels for one speckle, 1.1 and 0.96 pixels for the other one). For one speckle, Jaqaman et al's algorithm worked better than the particle filtering algorithm.

5.2 Measuring the Intensity Distribution of A Single Speckle

It has been shown that the fluorescent signal of a molecule could be fit with a Gaussian function [15, 18], whose peak shows the position of the molecule and whose standard deviation is limited by diffraction [18]. In order to investigate this, a set of 440 speckling points were chosen. The intensities of the pixels in a 15×15 patch around each speckle were used to derive the intensity distribution around the speckle, both in horizontal and vertical directions. These intensities were plotted versus their distance from the position of the speckle. It was found that the intensity distribution, in horizontal and vertical directions, could be fit with a Gaussian function (Figure 8). The fitted Gaussian functions can be displayed as:

$$I(x, y = y_0) = A_x \exp\left[-\frac{(x - x_0)^2}{2\sigma_x}\right] + B_x \quad (4)$$

$$I(x = x_0, y) = A_y \exp\left[-\frac{(y - y_0)^2}{2\sigma_y}\right] + B_y \quad (5)$$

where $A_x = 1.02 \times 10^4$, $x_0 = 0.94$, $\sigma_x = 2.53$, $B_x = 1.36 \times 10^4$, $A_y = 9.03 \times 10^3$, $y_0 = 1.17$, $\sigma_y = 2.79$ and $B_y = 1.36 \times 10^4$.

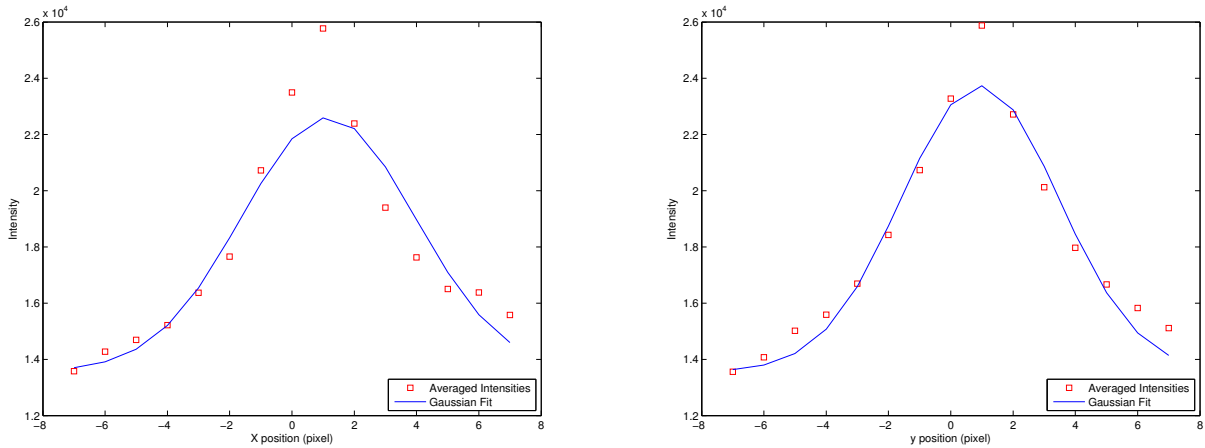


Figure 8: The intensity distribution around a single speckle, in both vertical and horizontal directions. The figure also shows the Gaussian fits given in equations 4 and 5.

In the above equations, x_0 and y_0 show the position of the intensity peak. Non-zero amounts of these values suggest that there is a difference between the given position of the speckle and the peak of the intensity. In other words, the values of x_0 and y_0 show that on average, the true position of each speckle is 1.17 pixels to the bottom and 0.94 pixels to the right of the positions given by the ground truth data. This means that there is 1 pixel error in tracking the speckles' positions by hand, a value comparable to the RMSE of the tracking algorithm.

The standard deviation σ can be calculated as $\frac{\sigma_x + \sigma_y}{2} = 2.66$ pixels or $2.66 \times 133 = 354$ nm. This is larger than the diffraction limit of 250 nm [8], in consistency with the fact that the standard deviation is limited by the diffraction limit [18]. The derived standard deviation is also larger than the one obtained in [18], which is ~ 200 nm. However, it may not be possible to make comparisons, because the width of the point spread function is dependant on the properties of the imaging device [15].

It can be inferred from the above equations that the intensity distribution of a single speckle can be approximated with a two dimensional Gaussian function:

$$I(x, y) = A \exp\left[-\frac{(x - x_0)^2}{2\sigma}\right] \exp\left[-\frac{(y - y_0)^2}{2\sigma}\right] + B \quad (6)$$

where A is the amplitude of the function, B is the background noise and σ is the standard deviation. This is consistent with the results given in [7, 18].

6 Discussion and Suggestions for Further Research

The particle filtering algorithm was used to track actin speckles. It was found that using the particle filtering technique, instead of the Kalman filter, decreased the average RMSE by 65 %, thus improving the tracking. This result was consistent with [21], which also demonstrated that the particle filtering technique was "a substantial improvement for detection and tracking".

Tracking was done for both blebbing and non-blebbing cells. It was found that in non-blebbing cells, using the Brownian motion model leads to better results, while in blebbing cells, the best results were obtained from downward motion. This suggests that interaction of the user with the algorithm, where the motion model could be chosen by the user depending on type of motion observed in the FSM movies, would improve the tracking results. In fact, in certain situations, such as blebbing, the motion of the speckles is towards a certain direction. In these situations, using a motion model which takes into account the direction of speckles' movement would lead to better results. In images where the motion of speckles is not unidirectional, Brownian motion may be a good choice for the dynamic model.

Although the new algorithm improved the results of tracking, the average RMSE was still considerable. In fact, the RMSE was 1.15 pixels or 152.95 nm, 28 times larger than the diameter of G-actin of about 5.5 nm [11]. To overcome the limit of resolution, caused by the diffraction of light [15], and localise the speckles with subpixel precision, the PSF function could be fit with a two-dimensional Gaussian function, as suggested in [7, 18]. In fact, in the particle filtering algorithm, after choosing the position of the particle with the highest weight as the subsequent location of the speckle, the intensity distribution around that point could be measured. By fitting a two-dimensional Gaussian function (equation 8) to the intensity distribution, x_0 and y_0 , the coordinates of the Gaussian peak, could be derived. When fitting the Gaussian, A and σ can float, while B , the background intensity, should have the fixed value of 1.36×10^4 , derived from fitting the Gaussian to the ground truth data. Since the peak of the PSF demonstrates the position of the point source [7], this would lead to subwavelength positioning of the particle.

It should be noted that the developed appearance model could be erroneous under particular conditions. In fact, when a brighter point is present near the speckle, the appearance model would give a higher weight to that point; hence, choosing the wrong point as the position of the speckle. Using other well-established appearance models, such as mean shift tracking, may further improve the results.

The work done in this research was focused on making the available particle filtering algorithm compatible with grayscale FSM images and improving its appearance model in order to track the speckles with more precision and accuracy. The motion model could also be improved. In fact, the parameters of the motion model were kept at their initial value. Modifying the values of these parameters, so that they better represent the dynamics of actin motion, may improve the tracking.

One of the features of Jaqaman et al's code is its ability to automatically detect the speckles, when they first appear in the image. The algorithm used in this research does not have this feature, requiring the user to provide it with the initial positions of the speckles. However, providing the algorithm with the first positions of speckles may be tedious and sometimes, impossible. In fact, some images have a high concentration of actin speckles. In addition, some of the speckles do not appear in the first frame, but in later frames. Thus, it may not be possible for the user to select all the speckles for tracking. Yet, it should be noted that even Jaqaman et al's algorithm was not able to detect all the speckles. Thus, implementing the ability of autodetection in the algorithm used in this research, along with its ability to

accept user-provided speckle positions, may ensure the detection of a higher number of speckles.

To conclude, the multi-modality of the FSM images obtained from the actin cortex prompt the need for using tracking algorithms which can cope with the complexity in these images. In this research, a particle filtering technique, which is well-equipped to deal with multi-modal situations, has been modified and utilised to track actin speckles. This led to a significant improvement in the results of tracking, compared to those obtained from the previously used algorithm. Although the RMSE between the tracked trajectories and ground truth data was less than the diffraction limit, the algorithm could still be improved to track the speckles with subwavelength precision. Several suggestions have been made to improve the algorithm. It is hoped that by applying the proposed changes to it, the tracking algorithm would be able to track the actin speckles with higher accuracy and precision, and thus, increase our insight into the structure and assembly of the actin cortex, as well as the actin dynamics.

7 Acknowledgements

I am grateful to both my supervisors, Dr Guillaume Charras and Dr Gabriel Brostow, for their help and support, and for acquainting me with two different worlds, the world of *cellular cytoskeleton* and the world of *computer vision*. I would also like to thank Marco Fritzsche, for sharing his experience on actin speckle tracking with me, as well as providing me with his FSM movies of the actin cortex.

A Appendix

The code for the speckle tracking algorithm, a sample video to run the algorithm on it, as well as the videos of the tracked speckles are available as supplementary materials. These supplementary materials have been handed it separately to CoMPLEX and will be provided upon request.

Below is the main part of the speckle tracking algorithm, as well as the algorithms which were developed to determine the sigmoid function and the PSF of a single speckle. All of the codes were written in MATLAB.

A.1 The Speckle Tracking Algorithm

```
%The main part of this code was developed by Dr Gabriel Brostow and  
%Cristina Garcia-Cifuentes. Some changes have been made in order to make  
%the code compatible with grayscale images. It is explicitly mentioned in  
%the code which parts were originally developed by me.
```

```
% Hard-coded data directory.  
% Contains folders named <VideoID> containing the PNG frames.  
global GDataDir  
GDataDir = '/Users/nargess/Desktop/SupplementaryMaterials/SampleVideo'  
close all;  
  
tmp = load('VideoID');  
videosToRun = tmp.videosToRun;  
  
for iVid = 1:numel(videosToRun)  
    % State here VideoId:  
    videoId = videosToRun{iVid};  
    % Load info  
    load([videoId, '_info'], 'startFrame', 'endFrame', 'dirName', ...  
        'frameNamePattern');  
    % Stored dirName might be wrong. Change it:  
    dirName = [GDataDir, '/', videoId, '/']; % where the pngs and GT data are.  
  
    flipVideo = false;  
  
    % Start tracking  
  
    %Different motion models can be chosen: Brownian 'brown', constant  
    %velocity 'const', downward 'bwd', upward 'fwd', leftward 'tkl' and  
    %rightward 'tkr'.  
    motion_model_strings={'brown', 'const','bwd'};  
    noise_levels = [ 0.1, 0.2, 0.5, 1, 2, 5 ];  
    for i=1:numel(motion_model_strings)  
        mmStr = motion_model_strings{i};  
        %for nl = noise_levels  
        for nl=0.1  
            close all;  
            track_video(startFrame, endFrame, dirName, frameNamePattern,...  
                mmStr, nl, flipVideo);  
        end  
    end  
end
```

end

A.1.1 Main Function of the Algorithm

The main function of the algorithm is as follows:

```
function track_video(iStartFrame, iEndFrame, sDirName, sFrameNamePattern, ...
    sMotionType, noiseLevel, flipVideo)
%TRACK_VIDEO Get input frames and interest points and track them.

% VideoID of the original video
videoID = strtok(sFrameNamePattern, '%');
% VideoID of the flipped video
videoID_flip = [videoID, '_FLIP_'];

% Input points and PNGs
% The PNGs are at the original video's directory: sDirName (includes '/')
pngDir = sDirName;
% The GT points are at the original video's Data directory
pointsDir = [sDirName, videoID, 'Data'];

% Output
if flipVideo
    ln = length(sDirName) - 1;
    sDirName_flip = [sDirName(1:ln), '_FLIP_']; % remove last '/' and ad '_FLIP_'
    if ~exist(sDirName_flip, 'dir')
        mkdir(sDirName_flip);
        fileattrib(sDirName_flip, '+w', 'g', 's');
    end
    sDirTracks = [sDirName_flip, '/', videoID_flip, 'Tracks', date]; % 'Tracks'
    %folder for the flipped video
else
    sDirTracks = [sDirName, videoID, 'Tracks', date];
end
if ~exist(sDirTracks, 'dir')
    mkdir(sDirTracks);
    fileattrib(sDirTracks, '+w', 'g', 's');
end

sDirTheseTracks = [sDirTracks, '/mm_', sMotionType, '_nl_', num2str(noiseLevel)];
if ~exist(sDirTheseTracks)
    mkdir(sDirTheseTracks);
    fileattrib(sDirTheseTracks, '+w', 'g', 's');
end

% Patch size
% -----
szPatch = [15, 15];
szPatchOffset = floor(szPatch./2);
% Get mesh grid centered at the template
xgv = (0:(szPatch(1)-1)) - szPatchOffset(1);
ygv = (0:(szPatch(2)-1)) - szPatchOffset(2);
[templX, templY] = meshgrid(xgv, ygv);

imgWidth = [];
imgHeight = [];
```



```

% Stopping condition
% -----
stopNframes = inf;
stopThreshold = 0.3;

% Number of particles per tracked object
% -----
numParticles = 100;

% Motion model
% -----
MotionModel = motion_models.init_motion_model( sMotionType, noiseLevel );
numDims = MotionModel.numDims;

% Tracked objects
objProto = struct(...
    'id', 0, ...
    'initPos', zeros(1,3,'uint16'), ...
    'template', zeros([szPatch(2), szPatch(1), 3], 'single'), ...
    'particles', zeros(numParticles, MotionModel.numDims, 'single'),...
    'weights', (1/numParticles)*ones(numParticles,1, 'single'),...
    'idxToPropagate', 1:uint16(numParticles), ...
    'appModel', [], ...
    'resample', false, ...
    'prediction', zeros(iEndFrame-iStartFrame, 3), ...
    'timeOfLife', uint16(0), ...
    'end', uint8(0), ...
    'debugMaxLife', uint16(0));
Objects = struct(objProto);
uniqueId = uint16(0);
countObj = uint16(0);

% Figures for visualization
szScr = get(0, 'ScreenSize');
hVisu = figure('Position', [5, 5, szScr(3)*0.9, szScr(4)*0.9]);
%jFrame = get(handle(gcf),'JavaFrame');
%jFrame.setMinimized(true); % to minimize the figure

for iFrame = iStartFrame:iEndFrame

    % Load current frame
    % -----
    frameName = sprintf(sFrameNamePattern, iFrame)
    Img = single(imread([pngDir, frameName, '.png']));
    Img2=imread([pngDir, frameName, '.png']);
    if isempty(imgWidth)
        imgWidth = size(Img,2);
        imgHeight = size(Img,1);
    end
    % FLIP?
    if flipVideo
        Img(:,:,:) = Img(:,(imgWidth:-1:1),:);
    end
    % Show
    set(0,'CurrentFigure',hVisu);

```

```

imshow(Img2);

%=====
%=====
% Particle filter iteration for each tracked object
% -----

for iObj = 1:countObj
    % Resample particles according to their likelihood
    % -----
    if Objects(iObj).resample
        Objects(iObj).idxToPropagate = resample_particles(...
            Objects(iObj).weights, numParticles);
    else
        Objects(iObj).resample = true;
    end

    % Propagate existing particles to current frame
    % -----
    newParticles = zeros(numParticles, size(Objects(iObj).particles, 2));
    for iSampl=1:length(Objects(iObj).idxToPropagate)
        idx = Objects(iObj).idxToPropagate(iSampl);
        newParticles(iSampl,:) = MotionModel.predict(...
            Objects(iObj).particles(idx,:));
    end
    Objects(iObj).particles = newParticles;
    clear newParticles;

    % Measure
    % -----
    if numParticles ~= length(Objects(iObj).weights)
        Objects(iObj).weights = zeros(numParticles,1);
    end
    for iPart = 1:numParticles

        %=====
        % Appearance model: Normalised Cross Correlation
        %-----

%         % Scaling and translation wrt initial template
%         sWH = Objects(iObj).particles(iPart,3:4) ./ szPatch;
%         scaleMat = diag(sWH);
%         translationXY = Objects(iObj).particles(iPart,1:2);
%         % Get patch from current image after transformation of template
%         % coordinates
%         Tform = maketform('affine', [scaleMat; translationXY]);
%         [imgX, imgY] = tformfwd( Tform, templX(:), templY(:) );
%
%         clear sWH scaleMat translationXY Tform;
%
%         % Check that the predicted location is a place we can really
%         % evaluate the likelihood.
%         inFrame = min(imgY) >= 1.0    &&    max(imgY) <= imgHeight && ...
%             min(imgX) >= 1.0    &&    max(imgX) <= imgWidth;
%         patch = zeros([szPatch]);

```

```

%         if inFrame
%             imgX = reshape(imgX, szPatch);
%             imgY = reshape(imgY, szPatch);
%             %patch(:, :, 1) = interp2(Img(:, :, 1), imgX, imgY, '*linear');
%             patch(:, :) = interp2(Img(:, :), imgX, imgY, '*linear');
%             %patch(:, :, 3) = interp2(Img(:, :, 3), imgX, imgY, '*linear');
%             ch = 2;
%             ncc = app_models.CalibNcc.patchSimilarity(patch(:, :), ...
%                 Objects(iObj).template(:, :));
%             Objects(iObj).weights(iPart) = Objects(iObj).appModel.ncc2proba(ncc);
%         else
%             Objects(iObj).weights(iPart) = 0.0;
%         end
%         Objects(iObj).particles(iPart, numDims) = Objects(iObj).weights(iPart);
%
%         clear imgX imgY patch;

%=====
% Appearance model: Assign weights to particles using the
% sigmoid function
% This part developed by Nargess
%-----
% Read the intensities of interior and exterior pixels
interior=zeros(3,3);
exterior=zeros(7,7);
x=uint16(Objects(iObj).particles(iPart,1));
y=uint16(Objects(iObj).particles(iPart,2));
interior=Img(y-1:y+1,x-1:x+1);
inFrame= y-3>=1.0 && y+3<=imgHeight&& x-3>=1.0 && x+3<=imgWidth;
if inFrame
    exterior=Img(y-3:y+3,x-3:x+3);
    % Calculate the difference between the average intensities
    % of interior and exterior pixels
    avginterior=mean2(interior);
    avgexterior=mean2(exterior);
    delta=avginterior-(((avgexterior*49)-(avginterior*9))/40);
    % Calculate the probability using the sigmoid function
    prob=sigmf(delta,[1 2108]);
else
    prob=0;
end
    Objects(iObj).weights(iPart)=prob;
    Objects(iObj).particles(iPart,numDims)=Objects(iObj).weights(iPart);
end

% Report our predicted position for this frame
% -----
Objects(iObj).timeOfLife = Objects(iObj).timeOfLife + 1;
[maxLike, maxIdx] = max( Objects(iObj).weights );
maxParticle = Objects(iObj).particles(maxIdx, :);
Objects(iObj).prediction(Objects(iObj).timeOfLife, :) = [maxParticle(1:2), maxLike];
% Stop condition
if maxLike < stopThreshold
    Objects(iObj).end = Objects(iObj).end + 1;
end

```

```

clear maxParticle

% Normalize weights
% -----
sumWeights = sum(Objects(iObj).weights);
if sumWeights == 0
    Objects(iObj).weights = ones(numParticles,1) ./ numParticles;
else
    Objects(iObj).weights = Objects(iObj).weights ./ sumWeights;
end
% Effective number of particles
%Neff = 1 ./ sum( Objects(iObj).weights .* Objects(iObj).weights );

end

%=====
%=====
% Stop tracking some objects
% -----
iObj = uint16(1);
while iObj <= countObj
    if Objects(iObj).end > stopNframes ...
        || Objects(iObj).timeOfLife >= Objects(iObj).debugMaxLife
        % Save sth somewhere
        annotation.save_track(sDirTheseTracks, Objects(iObj));
        % Delete
        Objects(iObj) = [];
        countObj = countObj - 1;
    else
        iObj = iObj + 1;
    end
end

end

%=====
%=====
% Display current tracks
% -----
set(0,'CurrentFigure',hVisu);
hold on
%=====
% Plot the tracked particles
%-----
for iObj = 1:countObj

    % plot particles
    plot(Objects(iObj).particles(:,1), Objects(iObj).particles(:,2), 'cx','MarkerSize',2)

    % plot predicted particle
    last = Objects(iObj).timeOfLife
    a=Objects(iObj).prediction(last,1)
    b=Objects(iObj).prediction(last,2)
    plot(Objects(iObj).prediction(last,1), Objects(iObj).prediction(last,2), ...
        'bo','LineWidth',1,'MarkerSize',5);

end

%=====
% Plot the ground truth points

```

```

% This part added by Nargess
%-----
    for i=iStartFrame:iFrame
        frameString = sprintf('frame%.6d', i); % hard-coded 6 digits
        fileNameFormat = [frameString, '_x%.5d_y%.5d.txt']; % har-coded 5 digit
        matchingData = dir([pointsDir, '/', frameString, '*.txt']);
        numGTD=numel(matchingData);
        hold on;
        for iData=1:numGTD
            GTD=importdata([pointsDir, '/', matchingData(iData).name], ' ', 1);
            numTrack=str2double(cell2mat(GTD.textdata));
            for iTrack=1:numTrack
                ifr=GTD.data(iTrack,3);
                if ifr==iFrame
                    x=GTD.data(iTrack,1);
                    y=GTD.data(iTrack,2);
                    plot(GTD.data(iTrack,1),GTD.data(iTrack,2),...
                        'ro','Linewidth',1,'MarkerSize',5);
                end
            end
        end
    end
end
%=====
% Save the tracks on videos
% This part added by Nargess
%-----
place=[sDirTheseTracks, '/', 'TrackImages', date];
if ~exist (place)
    mkdir(place);
    fileattrib(place, '+w', 'g', 's');
end
name=[place, '/Track', '_', num2str(iFrame)];
print ('-depsc', '-r300', name)
print ('-dpng', '-r300', name)

%-----
hold off
if countObj
    drawnow
end
%=====
% Add points to the list of objects to track in the next frame
% -----

[Points, maxLives] = annotation.search_points_to_track(pointsDir, ...
    sDirTheseTracks, iFrame, flipVideo, imgWidth);
if ~isempty(Points)
    numNewObjects = uint16(size(Points,1));
    % pre-allocation
    Objects((countObj + 1):(countObj + numNewObjects)) = struct(objProto);
    % init of each object
    for iPoint = 1:numNewObjects
        iObj = countObj + iPoint;
        Objects(iObj).id = uniqueId + iPoint;
        pos = Points(iPoint,:);
        Objects(iObj).initPos = uint16([pos, iFrame]);
    end
end

```

```

        % Correction if the patch is partly outside the image
        Objects(iObj).template= ...
            app_models.CalibNcc.get_pixels(Img, pos, szPatch);

        Objects(iObj).particles = MotionModel.init_particles(...
            [pos, szPatch], numParticles, imgWidth, imgHeight);

        Objects(iObj).appModel = app_models.CalibNcc(Img, pos, szPatch);

        Objects(iObj).debugMaxLife = maxLives(iPoint);

    end
    countObj = countObj + numNewObjects;
    uniqueId = uniqueId + numNewObjects;

    % Show
    set(0,'CurrentFigure',hVisu);
    hold on;
    plot(Points(:,1), Points(:,2), 'ro', 'LineWidth',2,'MarkerSize',10);
    hold off;
    drawnow
end

end% for frames

for iObj = 1:countObj
    % Save sth somewhere
    annotation.save_track(sDirTheseTracks, Objects(iObj));
end

try
    fileattrib([sDirTracks '/*'], '+w', 'g', 's');
catch ME1
end

end% fucntion

```

A.2 The Algorithm for Determining the Sigmoid Function

```

% This algorithm determines the parameters of the sigmoid function.
% For 136 speckling and 143 non-speckling points, the difference between
% the averaged intensities of interior and exterior pixels has been read
% and stored in 'Speckling_Intensity_19April.mat' and
% 'nonSpeckling_Intensity_19April.mat'.

```

```

% Load the intensities data
load('Speckling_Intensity_19April.mat');
load('nonSpeckling_Intensity_19April.mat');
mergedDelta=[SpecklesDelta,nonSpecklesDelta];

% Fit the data with a sigmoid function
x=mergedDelta;
y1=zeros(1,279);
y1(1,1:136)=1;

```

```

difference=zeros(20001,2);
i=1;
for c=-10000:1:10000
    diff=0;
    y2=sigmf(x,[1 c]);
    for j=1:279
        if y1(1,j)~=y2(1,j)
            diff=diff+1;
        end
    end
    difference(i,1)=c;
    difference(i,2)=diff;
    i=i+1;
end
% Choose the parameters which give the best fit
m=min(difference(:,2))
a=difference(:,2)==m;
finalc=mean2(difference(find(a),1));
%%
%=====
% Plot the sigmoid function
%-----
plot(x,y1,'r')
hold on;
plot(x,y2,'d')
xlabel('sigmf, P=[-1 c]')

%=====
%Calculate the averaged difference in intensities for speckling and
%non-speckling points
%-----
avgSpecklesDelta=mean2(SpecklesDelta)/65535;
avgNonSpecklesDelta=mean2(nonSpecklesDelta)/65535;

```

A.3 The Algorithm for Determining the PSF Function of a Single Speckle

```

% This algorithm determines the intensity distribution of a single speckle.
% The intensity distribution in a 15x15 patch around 440 speckling point
% has been read, averaged and stored in 'HIntensity.mat' and
% 'VIntensity.mat'.

```

```

% Read the intensity data
%-----
load('HIntensity.mat');
load('VIntensity.mat');

% Make directory to store the plots
%-----
dirName= '/Users/nargess/Desktop/'
histDir=[dirName,'Histograms']
if ~exist(histDir)
    mkdir(histDir);
    fileattrib(histDir,'+w','g','s');
end

```

```

% Determine the Gaussian fit in vertical direction
%-----
% Determine the background intensity
Vm=min(a)
c=a-Vm
% Find the Gaussian fit
[Vsigma,Vmu,VA]=mygaussfit(-7:7,c)

% Plot the intensities
plot(-7:7,a,'rs');
hold on;
plot(-7:7,VA*gaussmf(-7:7,[Vsigma Vmu])+Vm)
legend('Averaged Intensities','Gaussian Fit','location','SouthEast');
xlabel('X position (pixel)');
ylabel('Intensity');

% Save the plots and the parameters of the Gaussian fit
name=[histDir,'/Vplot_avg'];
print('-depsc','-r300',name);
print('-dpng','-r300',name);
hold off;
save('VGaussian.mat','Vsigma','Vmu','VA')

% Determine the Gaussian fit in horizontal direction
%-----
% Determine the background intensity
Hm=min(b)
c=b-Hm
% Find the Gaussian fit
[Hsigma,Hmu,HA]=mygaussfit(-7:7,c)

% Plot the intensities
plot(-7:7,b,'rs');
hold on;
plot(-7:7,HA*gaussmf(-7:7,[Hsigma Hmu])+Hm)
legend('Averaged Intensities','Gaussian Fit','location','SouthEast');
xlabel('y position (pixel)');
ylabel('Intensity');

% Save the plots and the parameters of the Gaussian fit
name=[histDir,'/Hplot_avg'];
print('-depsc','-r300',name);
print('-dpng','-r300',name);
hold off;
save('HGAussian.mat','Hsigma','Hmu','HA')
%%

```


References

- [1] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Molecular Biology of the Cell*. Garland Science, 5th edition, 2008.
- [2] Bruce Alberts, Dennis Bray, Karen Hopkin, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Essential Cell Biology*. Garland Science, 3rd edition, 2010.
- [3] D Bray and JG White. Cortical flow in animal cells. *Science*, 239(4842):883–888, 1988. doi: 10.1126/science.3277283.
- [4] Guillaume Charras. A short history of blebbing. *Journal of Microscopy*, 231:466–478, 2008. doi: 10.1111/j.1365-2818.2008.02059.x.
- [5] Guillaume Charras and Ewa Paluch. Blebs lead the way: How to migrate without lamellipodia. *Nature Reviews Molecular Cell Biology*, 9:730–736, 2008. doi: 10.1038/nrm2453.
- [6] Guillaume T. Charras, Chi-Kuo Hu, Margaret Coughlin, and Timothy J. Mitchison. Reassembly of contractile actin cortex in cell blebs. *The Journal of Cell Biology*, 175(3):477–490, 2006. doi: 10.1083/jcb.200602085.
- [7] Michael K. Cheezum, William F. Walker, and William H. Guilford. Quantitative comparison of algorithms for tracking single fluorescent particles. *Biophysical Journal*, 81(4):2378–2388, 2001. doi: 10.1016/S0006-3495(01)75884-5.
- [8] Gaudenz Danuser and Clare M. Waterman-Storer. Quantitative fluorescent speckle microscopy of cytoskeleton dynamics. *Annual Review of Biophysics and Biomolecular Structure*, 35(1):361–387, 2006. doi: 10.1146/annurev.biophys.35.040405.102114.
- [9] Darius M. Dziuda. *Data Mining for Genomics and Proteomics: Analysis of Gene and Protein Expression Data*. John Wiley Sons, 2010.
- [10] Kathryn M. Eisenmann, Elizabeth S. Harris, Susa M. Kitchen, Holly A. Holman, Henry N. Higgs, and Arthur S. Alberts. Dia-interacting protein modulate formin-mediated actin assembly at the cell cortex. *Current Biology*, 17:579–591, 2007. doi: 10.1016/j.cub.2007.03.024.
- [11] Monika Fritz, Manfred Radmacher, Jason P. Cleveland, Miriam W. Allersma, Russell J. Stewart, Ralph Gieselmann, Paul Janmey, Christoph F. Schmidt, and Paul K. Hansma. Imaging globular and filamentous proteins in physiological buffer solutions with tapping mode atomic force microscopy. *Langmuir*, 11(9):3529–3535, 1995. doi: 10.1021/la00009a040.
- [12] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113, 1993.
- [13] Chiharu Higashida, Shiro Suetsugu, Takahiro Tsuji, James Monypenny, Shuh Narumiya, and Naoki Watanabe. G-actin regulates rapid induction of actin nucleation by mdia1 to restore cellular actin polymers. *Journal of Cell Science*, 121:3403–3412, 2008. doi: 10.1242/jcs.030940.
- [14] Khuloud Jaqaman, Dinah Loerke, Marcel Mettlen, Hirotaka Kuwata, Sergio Grinstein, Sandra L Schmid, and Gaudenz Danuser. Robust single-particle tracking in live-cell time-lapse sequences. *Nature Methods*, 5(8):695–702, 2008. doi: 10.1038/nmeth.1237.
- [15] Erik Meijering, Ihor Smal, and Gaudenz Danuser. Tracking in molecular bioimaging. *Signal Processing Magazine, IEEE*, 23(3):46–53, 2006. doi: 10.1109/MSP.2006.1628877.
- [16] Erik Meijering, Oleh Dzyubachyk, Ihor Smal, and Wiggert A. van Cappellen. Tracking in cell and developmental biology. *Seminars in Cell and Developmental Biology*, 20(8):894–902, 2009. doi: 10.1016/j.semcdb.2009.07.004.
- [17] Takushi Miyoshi, Takahiro Tsuji, Chiharu Higashida, Maud Hertzog, Akiko Fujita, Shuh Narumiya, Giorgio Scita, and Naoki Watanabe. Actin turnover-dependent fast dissociation of capping protein in the dendritic nucleation actin network: Evidence of frequent filament severing. *The Journal of Cell Biology*, 175(6):947–955, 2006. doi: 10.1083/jcb.200604176.

- [18] W.E. Moerner. Microscopy beyond the diffraction limit using actively controlled single molecules. *Journal of Microscopy*, 2012. doi: 10.1111/j.1365-2818.2012.03600.x.
- [19] Ewa Paluch, Jasper van der Gucht, and Cecile Sykes. Cracking up: Symmetry breaking in cellular systems. *The Journal of Cell Biology*, 175(5):687–692, 2006. doi: 10.1083/jcb.200607159.
- [20] Dr Simon J. D. Prince. *Computer Vision: Models, Learning, and Inference*. Cambridge University Press, <http://www.computervisionmodels.com>, 2012.
- [21] Ihor Smal, Wiro Niessen, and Erik Meijering. Particle filtering for multiple object tracking in molecular cell biology. In *Nonlinear Statistical Signal Processing Workshop, 2006 IEEE*, pages 129–132, 2006. doi: 10.1109/NSSPW.2006.4378836.
- [22] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, London, 2011.
- [23] Takahiro Tsuji, Takushi Miyoshi, Chiharu Higashida, Shuh Narumiya, and Naoki Watanabe. An order of magnitude faster arp1-associated actin disruption than nucleation by the arp2/3 complex in lamellipodia. *PLoS ONE*, 4(3), 2009. doi: 10.1371/journal.pone.0004921.
- [24] Naoki Watanabe. Inside view of cell locomotion through single-molecule: Fast f-/g-actin cycle and g-actin regulation of polymer restoration. *Proceedings of the Japan Academy, Series B*, 86(1), 2010. doi: 10.2183/pjab.86.62.
- [25] Naoki Watanabe and Timothy J. Mitchison. Single-molecule speckle analysis of actin filament turnover in lamellipodia. *Science*, 295:1083–1086, 2002. doi: 10.1126/science.1067470.
- [26] Clare M. Waterman-Storer, Arshad Desai, J. Chloe Bulinski, and E.D. Salmon. Fluorescent speckle microscopy, a method to visualize the dynamics of protein assemblies in living cells. *Current Biology*, 8(22):1227–1230, 1998. doi: 10.1016/S0960-9822(07)00515-5.
- [27] Steven J. Winder and Kathryn R. Ayscough. Actin-binding proteins. *Journal of Cell Science*, 118: 651–654, 2005. doi: 10.1242/jcs.01670.