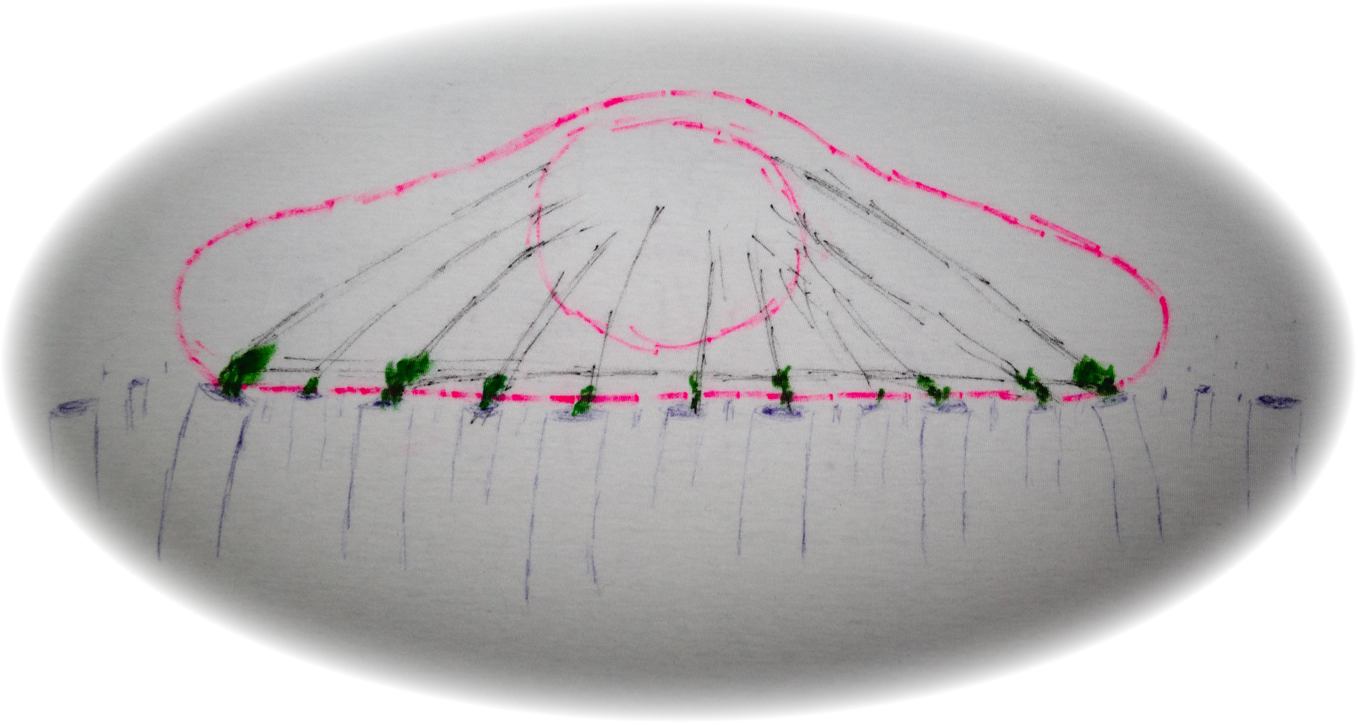
Probing actin cytoskeleton dynamics using a micropillar array

Master thesis
January 25, 2013

W. Pomp, BSc



Supervised by:

Ir. H van Hoorn
Prof. dr. T. Schmidt
Leiden University

This report is written based on a six month graduation project in the group 'Physics of Life Processes' for the master 'Research in Experimental Phyics' at Leiden University by Wim Pomp, BSc.

Email: wimpomp@gmail.com

Supervisors for this project were:

Ir. H. van Hoorn
Prof. dr. T. Schmidt

Front cover image: Graphical representation of a cell with actin and focal adhesions on pillars.

Abstract
Environmental cues regulate cell fate and behavior. Cells use force sensors while exerting forces on the environment to probe properties such as the rigidity of the environment. We built an autofocus system onto a spinning disk confocal microscope. This enables us to capture time-lapse movies of cells expressing fluorescent actin in an automated manner. An inverted pillar array substrate was used to measure forces the cells exert on the substrate. In this way we were able to confirm and quantify a theory suggesting that cells use the leading edge of the cell to pull themselves forward. Furtermore, in more than 90% of the observed cells distinct cortical actin stress-fibers were found. These cases were checked against active gel theory assuming a homogeneous contractilty. However, in our experiments, forces described by this model are not significant and we propose an extension to the active gel theory incorporating stress-fiber force exertion. Our new model agrees well with our experiments and shows that forces mediated by stress-fibers are 14 times higher than forces generated by homogeneous contractility.

Contents

1 Introduction

Understanding mechanisms governing cell fate is crucial in several medical applications. It is known that stem cell fate depends on various external cues. It has been shown that stem cell differentiation, for example, depends on the rigidity of the surrounding tissue [1]. Also, the anisotropy of actin stress-fiber alignment, determining cell fate, depends on the rigidity of the surroundings [2]. The actin network plays a central role in generating and mediating forces, exerted on the surroundings of the cell. By exerting forces the cell can probe the surrounding tissue, and estimate properties like the rigidity of the surrounding tissue.

1.1 The cytoskeleton and extracellular matrix

The basic unit of every organism is the cell [4]. While there are major differences between different cells, many features are common to most cells. For example, all cells have a cell membrane. Also, most cells have a cytoskeleton, providing rigidity to the cell.

The cytoskeleton is a network of three different kinds of proteins, of which two can be seen in figure 1. The microtubules are the largest filaments with a diameter of 25 nm. Microtubules are important in transport in the cell and in cell division. The 10 nm diameter intermediate filaments constitute a large family, occurring in different places in the cell, for example as a network of lamin in the nucleus. The protein that forms the smallest of the cytokeletal filaments is actin. A single actin filament has a diameter of 5 to 9 nm.

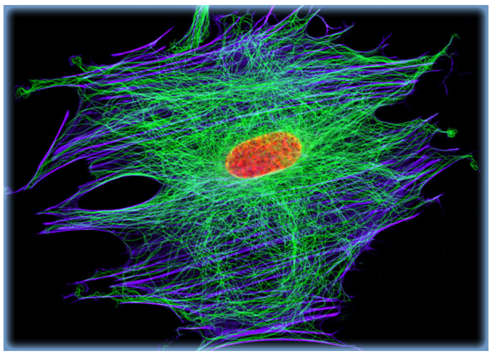


Figure 1: The cytoskeleton [3]. In this image the microtubules are colored blue, actin green, and the nucleus red.

Eukaryotic cells have a nucleus somewhere near the center. Next to the nucleus is a microtubule organizing center from which microtubules extend outward to the edge of the cell. Throughout the whole cell, but mainly at the cortex extends an actin network. Sometimes multiple actin fibrils are bundled together into actin fibers that can even span the whole cell. At leading edges of the cell protrusions filled with a dense actin network exist. The flat edges of these lamellae are called lamellipodia and the actin network in these protrusions is highly dynamic.

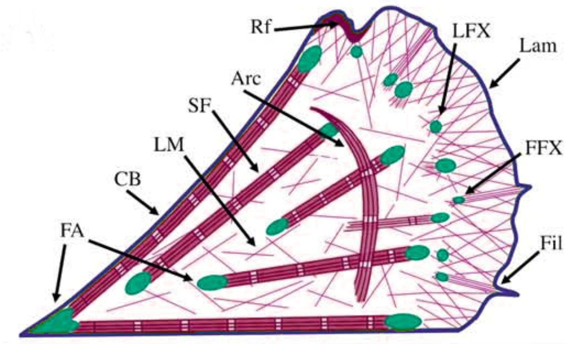


Figure 2: Actin network with focal adhesions [5]. The cell membrane is blue, actin is red and the focal adhesions are green.

In a tissue outside the cell consists of a network which we call the extracellular matrix (ECM). Cells can attach to the ECM and use it to migrate through tissues. The proteins forming the ECM, for example collagen and fibronectin, are produced by the cells in the ECM. Migrating cells often break down the ECM in front, and regenerate it behind the cell. The connection between the ECM and the cytoskeleton is provided by integrins and focal adhesions (FA). Integrins are transmembrane proteins, providing anchor points for the ECM outside the cell, and binding to FAs inside the cell. A focal adhesion is a large complex, consisting of many different proteins. Once formation of a focal adhesion at an integrin has started actin can bind to it.

Part of the actin network is enriched with myosin motors which slide actin filaments alongside each other, contracting parts of the actin network. Integrins in the cell membrane transmit forces generated by the actin myosin network to the ECM outside the cell. Forces generated in different parts of the cell can be transmitted to the cell membrane by actin fibers. Naturally, thicker actin fibers should be able to sustain bigger forces, the thick stress fibers being the strongest. This leads to the hypothesis that in a cell the majority of the forces is mediated by stress fibers, and in a polarized cell this should be clearly visible because of a lot of force exterted in the direction of the stress fibers.

Arrays of micropillars have been used to vary substrate stiffness without modifying other substrate properties [6], showing the dependency of cell behaviour on global substrate stiffness. At the same time, a pillar array can be used to measure forces [7]. Theories describing cell migration predict that cells use a protrusion of the membrane filled with actin to probe the environment and to pull themselves forward [8].

We measured these forces over time and verified that indeed the leading edge of the cell exerts most of the force.

Forces in the cytoskeleton generate tension which affects the shape of a cell. This behavior has been described in the active gel theory by modeling the cell as a homogeneous gel of contractile actin-myosin motors [9]. As we will show, a majority of the forces generated by the actin network are mediated by stress-fibers; thick cross-linked bundles of many actin filaments. Furthermore, we show a dependency between the orientation of stress-fibers in the cell relative to the edge of the cell and forces. We therefore propose an extension to the active gel theory, incorporating forces mediated by stress-fibers.

2 Theory

2.1 Cell dynamics

Integrins and focal adhesions play a major role in the process of cellular force sensing. Not only do they provide the connection between the cell and its surroundings, but integrins also are particularly good at transmitting forces through the membrane. A changing force on one side of an integrin dimer changes the conformation of the pair, which in turn also changes the conformation of the integrins at the other side of the membrane. Furthermore, several proteins inside FAs, for example talin and zyxin as well as the focal adhesion kinase and p130Cas proteins, are thought to act as force sensors, exposing a binding site when stretched by a force [10]. This mechanism is probably also involved in the growth of focal adhesions, as recent reseach suggests that proteins are recruited by a focal adhesion when a connected actin filament exerts a force on it [8].

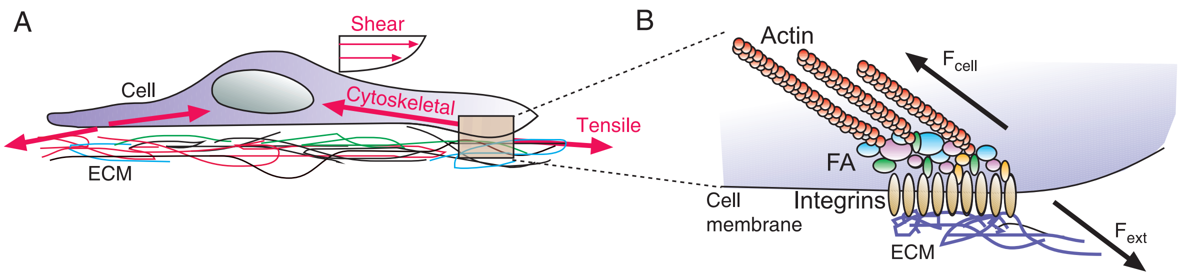


Figure 3: Focal adhesions [11]. **A.** A cell sitting on the ECM, the cell exerts a force on the ECM and the ECM exerts a force on the cell. **B.** The forces are transmitted from the actin network to the ECM and vice versa through focal adhesions attached to integrins.

The cytoskeleton forms a skeleton for the cell, but for a cell to move, the cytoskeleton also has to be flexible. The main feature that adds to the flexibility of the cytoskeleton is the instability of the cytoskeleton. It is constantly depolymerizing and polymerizing everywhere in the cell. In this way the cell can rapidly change its shape. Several proteins are employed in this process, for example to prevent or promote binding to either of the ends of actin filaments. Since actin monomers undergo a conformational change when they bind to one of either ends of a filament, the binding affinity at either ends is different [4]. The fast growing end is called the plus or barbed end and is always pointed towards the cell periphery. The minus end is always pointed to the center of the cell and usually the binding affinity there is so low that it is depolymerizing. This process of polymerization at the plus end and at the same time depolymerization at the minus end is called tread-milling.

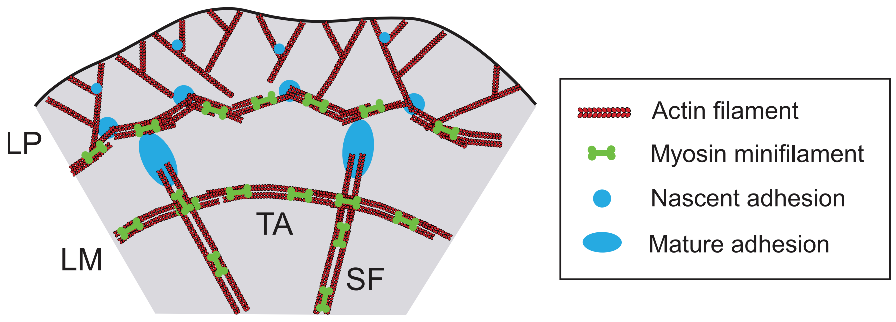


Figure 4: Actin network [8]. The lamellipodium, near the edge of the cell, has a heavily branched actin network, with nascent adhesion. More to the center of the cell myosin contracts the actin into more rigid fibers. This force also matures adhesions into focal adhesions. LP: lamellipodium, LM: lamella, SF: stress-fiber.

The actin network in the bulk of the cell is decorated with myosin. Myosin is a motor-protein, and as such can pull one actin filament along another. By doing this the actin network in the bulk forms thicker actin fibers. Also, this contractile network pulls on the heavily branched actin network inside the lamellipodia. These lamellipodia are formed by polymerizing actin which pushes the membrane outward, forming a protrusion.

On the integrins in this membrane, adhesions start to form, as shown in figure 5A(1). The actin network binds to those nascent adhesions. The contractile actin network in the bulk of the cell pulls on the network inside the lamellipodia. Because of the tension this generates on the adhesions in the lamellipodia, the adhesions start to grow into focal adhesions. At the same time the integrins, pulled on by focal adhesions, pull on the environment outside the cell. If the cell then for some reason does not move in that direction, the lamellipodia often become ruffles, the actin depolymerizes and the extended membrane travels back in a wave-like motion. However, when the connection between the integrins and the environment are rigid enough, the contents of the cell move forward and the focal adhesions move into the lamellae (figure 5A(3)). This process can be continuous, such that in fast moving cells such as fish keratocites there is always a lamellipodium, extending at the leading edge, and disappearing in the bulk at the rear [12].

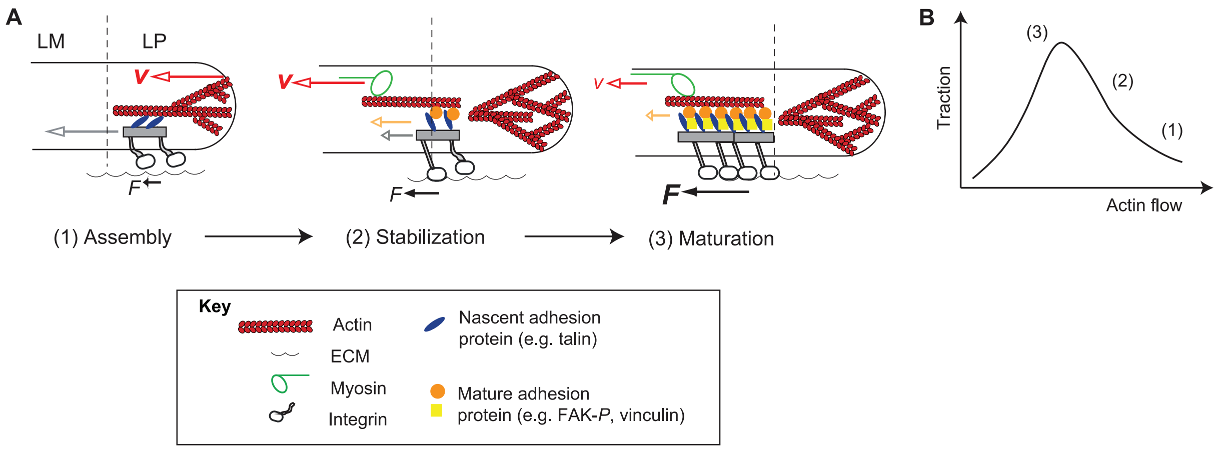


Figure 5: Adhesion dynamics [8]. **A** Adhesion proteins start to grow in the lamellipodium (1), under tension they mature into focal adhesions (3). **B** Tractionforce versus retrograde flow of the treadmilling actin in the lamellipodium. The flow is a measure for position in the lamellipodium, with the flow being biggest near the cell front of the cell.

Growth of adhesions is governed by forces exerted on the adhesions. Force-sensing proteins can recruit more adhesion proteins to the adhesion. Forces on the adhesions are exerted by the contractile actin-myosin gel connecting the leading edge of the cell to the cell contents in the bulk of the cell. At some point the focal adhesions themselve move into the bulk of the cell. Here, the contractile actin exerts less or no force on them. Under this decreased force the focal adhesions start to disassemble and eventually will disappear completely. In figure 5B, the force on a adhesion is plotted versus the flow of treadmilling actin, which is a measure for the position in the lamellipodium. We see that adhesions start at a position with high actin flow and when maturing the force on the adhesions increases and the actin flow decreases. Eventually, the flow still decreases, but the traction forces decreases again when the focal adhesion disassembles. Based on this we can hypothesize that when cells migrate they use the leading edge of the cell to pull themselves forward.

2.2 Active gel and stress-fibers

The shape and structure of a cell is mainly determined by the actin network, of which a part is enriched with myosin. The myosin-enriched actin network is contractile and so a cell can be described as an active, contractile gel. This has been done in previous work [13,14]. This is a purely physical description in which it is assumed that the gel exerts a uniform force, perpendicular to the membrane, on the membrane. Using this description, it can be shown from minimizing the energy functional for the cell contour:

$$E = \int \sigma dA + \int \lambda dl \tag{1}$$

that if the line tension (λ) and surface tension (σ) along a stress-fiber next to a cell boundary are constant, then the stress-fiber follows a circular path. The surface tension can then be calculated as:

$$\sigma = \frac{\lambda}{R} \tag{2}$$

In a two-dimensional picture, like in figure 6, λ and σ are measured in units of force and force per length respectively. The line-tension can be measured as a force at the ends of the stress-fiber. It is convenient to view the surface tension as a measure of the contractility of the active gel. However, the model leading to equation 2 only works if the network exhibits no resistance against compression, and the surface tension is constant and perpendicular to the stress fiber (figure 7A), similar to a uniform pressure.

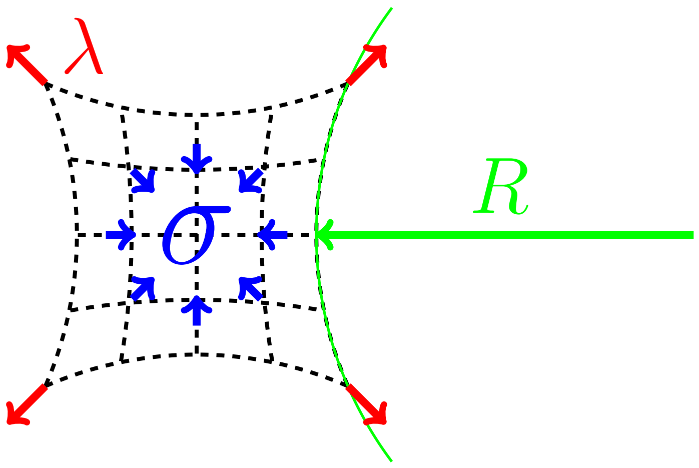


Figure 6: A contractile actin network (black) has edges which are circular in shape with radius R (green). The line-tension λ (red) can be measured as a force on the ends of a stress-fiber running along the edge. The contractility σ (blue) of the gel can be calculated from the line-tension and the radius.

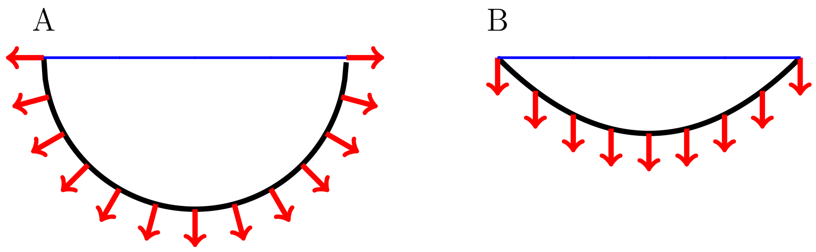


Figure 7: **A.** When forces (red arrows) are perpendicular to a fiber (black) and have the same magnitude, then the fiber will assume a circular shape. **B.** If the forces all point in the same direction, the fiber will assume the shape of a parabola.

If the actin does not exert a uniform pressure on the edge-fibers, then the shape of the edge-fiber will deviate from circularity. If for example the surface tension on the edge-fiber is not perpendicular to the fiber, but on every point on the fiber points in the same direction, then the fiber will take the shape of a parabola (figure 7B). This conformation will make equation 2 incorrect, because the shape of the fiber is no longer circular. However, we can always approximate a parabola with a circle and thus define a radius. In reality the shape of the fiber will probably be somewhere between circular and parabolic because of the contributions of the active gel and basal stress-fibers. Therefore, we propose a new model which adds the force that basal stress-fibers exert on stress-fibers along the edge of the cell to the active gel description.

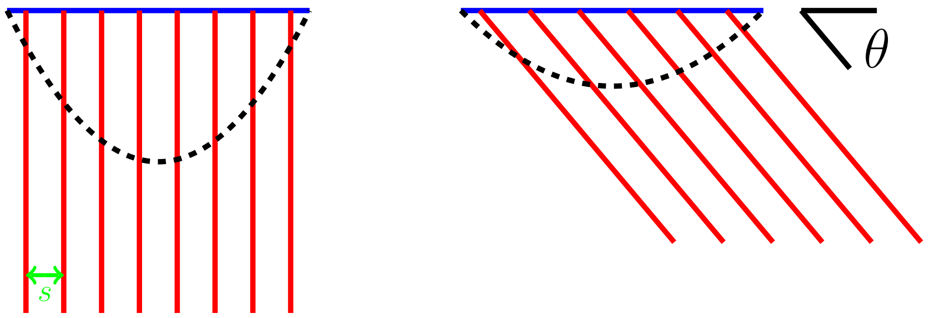


Figure 8: The density of stress-fibers (red) attached to the edge (blue) depends on the spacing s between stress-fibers and on the angle between the stress-fibers and the edge. So the shape the edge is then pulled into (black) also depends on this angle.

To account for the contribution of basal stress-fibers to the stress on the cortical fibers we assume that the shape of the cortical fiber does not deviate much from circularity, so that equation 2 is still a good approximation. If we assume that stress-fibers are uniformly spaced (spacing s) and hit the edge stress-fiber at a common angle θ, as depicted in figure 8, then the density n of attachments of basal stress-fibers to the edge fiber is given as:

$$n = \frac{1}{s} \sin\theta \tag{3}$$

Equation 3 is exact if the edge fiber assumes the shape of a straight line between the attachment points. When the shape is not a straight line two things happen. Depending on the angle of the basal stress-fibers, more stress-fibers attach to the edge fiber, increasing the surface tension. At the same time the fiber length is longer, decreasing the surface tension. These two effects roughly cancel each other, so that equation 3 can be used for most edge fibers in a cell. If we then assume that the stress f per basal stress-fiber is roughly the same for every basal stress-fiber, an approximation to the surface tension due to basal stress-fibers is:

$$\sigma_{SF} = \frac{f}{s} \sin\theta \tag{4}$$

Using equation 2 again and at the same time combining contributions from the active gel and stress-fibers we arrive at:

$$\sigma = \frac{\lambda_{AG}}{R} \left[1 + \frac{\lambda_{SF}}{\lambda_{AG}} \sin\theta\right] \tag{5}$$

Where λ_{AG} and λ_{SF} are contributions to the force from the active gel and basal stress-fibers respectively.

A tension-elasticity model has been proposed [14] that incorporates elastic fibers via an one-dimensional elastic modulus EA which is the product of a three-dimensional elastic moduls and a cross-sectional area; and a resting length $L_0 = \alpha d$ modifying the tension:

$$\sigma_{AG} = \frac{EA}{R} \left[\frac{2R}{\alpha d} \arcsin\left(\frac{d}{2R}\right) - 1\right] \tag{6}$$

It has been used to explain different radii R of circular parts of the cell membrane for different spanning distances d. However, this model does not incorporate the importance of basal stress-fibers. It could be expanded using equation 4 to also describe basal stress-fibers. However, both elastic modulus and resting lengths are unknown in our experiments, so we use equation 5 as an approximation instead.

3 Materials & Methods

3.1 Confocal microscopy

All experiments were done using a Zeiss Axiovert 200 inverted microscope and a Zeiss plan-apochromat 100× 1.4 oil imersion objective. The microscope is equiped with a Yokogawa spinning disk unit and a Andor iXon+ camera. A 100 mW 405 nm laser from CrystaLaser and a 100 mW 561 nm Cobolt laser were used for fluorescense imaging. For automatic sample positioning a Marzhaüser Scan IM 120×100 stage and a PI piezo objective postioning system were used.

A warm water circulation system flowing heated water around the objective and the sample kept the sample at 37°C. The microscope stage insert and heated sample-holder were produced such that possible movements of the sample were reduced to a minimum. This allows us to image different spots on the sample repeatedly.

3.2 Autofocus and drift correction

Some of our experiments involve time-lapse movies, but due to the shallow focus depth of our objective and drift, focus can easily be lost. To overcome this problem we built an auto-focus system using an IR-laser to keep the distance between objective and microscope slide constant.

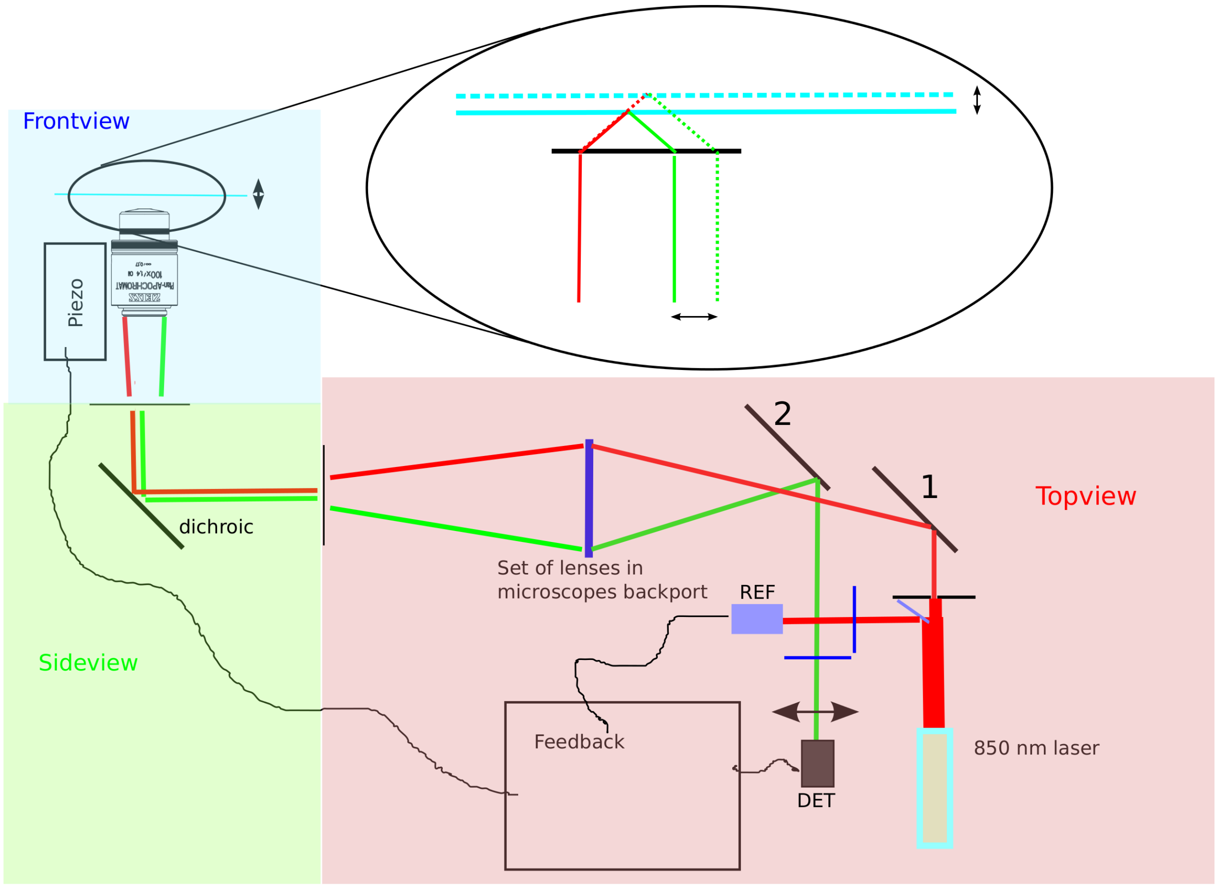


Figure 9: Auto focus setup. Laser-light reflects from the sample, such that the amount of light that hits the detector depends on the distance between sample and objective. A feedback system is then used to keep this distance constant. DET: Thorlabs detector, REF: reference detector.

As shown in figure 9, we use an 850 nm laser (Axiz 850 nm, 5 mW) and let the light enter the optical path via the backport of the microscope and a dichroic mirror (Chroma T800dcspxr). The light passes the objective off-axis so that it hits the microscope slide at an oblique angle. At the glass-sample interface about 4% of the light is reflected back. The entry point into the objective now depends on the distance between objective and microscope slide, and thus the further path of the light. Depending on this, some part of the light hits a detector (Thorlabs PDA36A). The output from this detector and that of a reference detector (BPX65) measuring laser power is used as a feedback value. Keeping the ratio between the outputs of these detectors constant means keeping the distance between objective and sample constant and thus keeping the object in focus.

Software for the auto-focus system was written in Labview (National Instruments Labview 2011) and Python. The Labview program named ´General Control´ is connected to the hardware by a National Instruments PCI6250 data acquisition card. It controls the auto-focus system with positioning stage and a substrate stretcher not used in the experiments in this thesis. General Control interfaces with Andor IQ 2.6 microscope software by means of TTL triggers. In this way the user of the microscope is able to specify an imaging protocol in Andor IQ that controls the autofocus and positioning stage in an automated way.

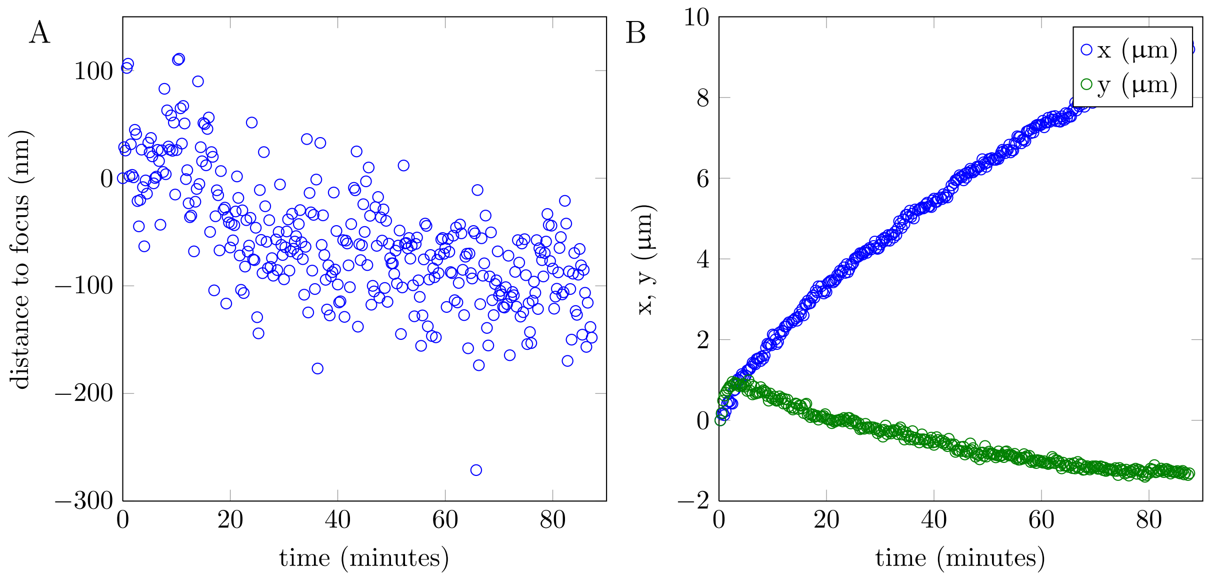


Figure 10: A. Drift from focus in about 1.5 hours. Tested using fluorescent beads stuck to a microscope slide, submersed in water, heated to 37 °C. **B.** Trajectory of drift in the horizontal plane in the same 1.5 hours.

As shown in figure 10A, our autofocus system is able to stay within 200 nm from the chosen focus, well within the focusdepth of the objective of about 1 μm. The data shown in figure 10A is collected using suboptimal gain settings, resulting in a oscillation around the actual focus. Using optimized gain settings results in a 100 nm accuracy without drift. This data is generated using beads stuck to the microscope slide as a test sample. The slide was submersed in water and heated to 37° C while measuring. Afterwards the intensity of the beads was compared to the intensity of the beads in images at known distances from focus.

Figure 10B shows the trajectory of the drift in the horizontal plane, it shows that there is drift in these directions too. Since this drift is only a few percent of the field of view, we compensate for this afterwards by doing a cross-correlation between frames. In our experiments with cells and pillars we do have for each time point a frame with pillars and a frame with actin. Because the cell image is dynamic we can only use the correlation between subsequent frames. To prohibit errors summing up we also correlate each frame with pillars with the first frame. The pixel accuracy thus acquired is enough to track the cell and pillars over time.

3.3 Sample preparation

3.3.1 Pillar arrays

To measure forces we used a PDMS pillar array with pillars of 2 µm diameter and 6.9 µm height in a hexagonal pattern with a 4 µm center-to-center distance. An electron micrograph is shown in figure 11a. Flanking the array on two sides are 50 µm high spacers (figure 11b), which allow for 43 µm space between pillars and glass when the array is upside down on a microscope slide for imaging. To make this array we have a silicon wafer as a mold. PDMS and curing agent where mixed in a 1 to 10 ratio and then degassed. After pouring the PDMS on the silanized wafer the PDMS is degassed again and cured for 16 hours at 110°C. After curing the 10 by 10 mm pillar arrays are peeled off of the wafer and excess PDMS is cut off. The surface of the PDMS is then activated by UV-ozone treatment for 30 minutes.

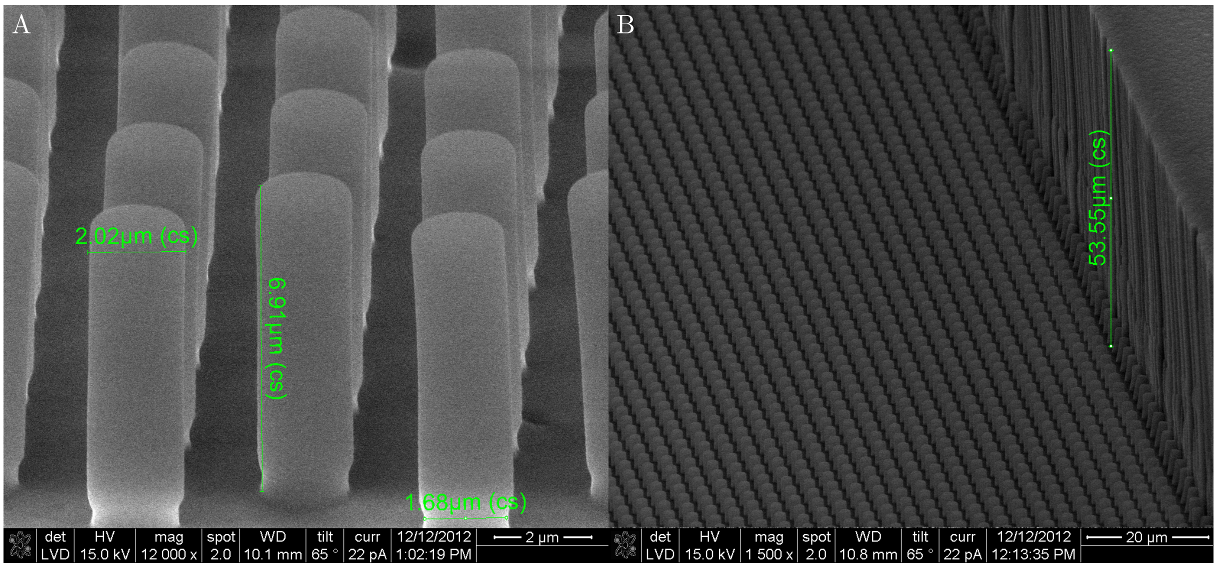

A
2.02µm (cs)
6.91µm (cs)
1.68µm (cs)
det LVD HV 15.0 kV mag 12 000 x spot 2.0 WD 10.1 mm tilt 65 ° curr 22 pA 12/12/2012 1:02:19 PM 2 µm
B
53.55µm (cs)
det LVD HV 15.0 kV mag 1 500 x spot 2.0 WD 10.8 mm tilt 65 ° curr 22 pA 12/12/2012 12:13:35 PM 20 µm

Figure 11: Electron micrographs of a pillar array. **A.** Dimensions of a pillar are shown. **B.** The spacer of about 50 µm high keeps some space between the pillars and the microscope slide.

A fibronectin solution consisting of 50 µg/ml fibronectin and 10 µg/ml fibronectin conjugated with Alexa-405 was prepared. On a flat piece of PDMS somewhat smaller than 10 by 10 mm, 40 µl of this solution was deposited. The fibronectin was allowed to sink to the PDMS stamp for one hour before wicking off the liquid and gently washing the stamp with pure water (Merck Millipore MilliQ). Then the stamp was applied to the ozone treated pillar array for 15 minutes. This procedure makes sure only the tops of the pillars get coated with fibronectin. To block all other surfaces the pillars arrays were submersed in a solution of 0.2% pluronic f127 in PBS, and the stamp was removed. After 30 minutes the pluronic was washed out by PBS.

3.3.2 Cell culture

For our experiments we used NIH 3T3-Fibroblasts with a stable expression of mCherry-LifeAct. This complex binds to actin and enables us to visualize the actin network. The cells were kept in medium (DMEM high glucose without phenol red or glutamine, with 10% NBS, 100 µg/ml pen/strep and 10% glutamax) in a 37° C, 7% CO_2 incubator and passaged twice a week. At least four hours before an experiment the cells were seeded in a 1:20 ratio onto a pillar array submersed in medium in a six-well plate well (962 mm^2 area). This gives the cells time to properly attach to the pillar array which is needed because unattached cells sink away from the pillar array which we put upside down on the microscope.

3.4 Stress fiber and pillar deflection analysis

3.4.1 Deflections and forces

We use image analysis in Matlab (Mathworks Matlab R2012a) to determine the deflection of the top of a pillar from which we can calculate the force on the pillar. The force-deflection relationship is determined using finite element analysis (figure 12). If the contrast of the fluorescent pillar-tops to the background is good enough we can transform the microscope image into a black-and-white image. From this image the centers of the pillars can be found and at the same time features that do not represent pillars can be removed. From this list of pillar coordinates the original grid before deflection can be reconstructed given that most of the pillars are not deflected. Because our array is about 50 µm above the microscope slide we have to compensate some aberrations. This is done by fitting the smallest 90% of the deflections respective to a perfect grid on a 5th order polynomial function. The reference grid is then adapted to this polynomial. In a time-series the average polynomial over all frames is used to correct the reference grids. Individual pillars can be tracked over time after drift in the horizontal plane is corrected which is described in section 3.2. Pillars in consecutive frames are then identified by the shortest distance between the pillars.

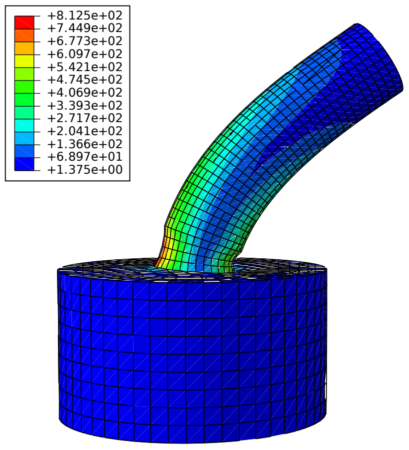


Figure 12: Finite element analysis of one PDMS pillar gives a force-deflection relation. Stress is represented by color, with hotter colors representing bigger stress.

3.4.2 Stress-fiber analysis

To find stress-fibers we made a script that first converts a grayscale image into a black-and-white image using a low-pass filtered image and a threshold one standard deviation above the mean. Now the image is cleaned and all small features are filtered out. Using standard Matlab functions to make a skeleton representation of the image, end- and branch-points can be found. Now some fibers are connected depending on their relative orientation and are given a unique number. Using these numbers, a list is generated listing the locations, sizes and orientations of the stress-fibers. This procedure finds the fibers at the edges of cells and also the brightest fibers in the bulk of the cell.

After stress-fibers are found further analysis reveals radii of the circular parts of stress-fibers. This involves manual selection from the list of the stress-fibers generated by earlier described script. The selected stress-fibers are fitted to circle shapes, and the forces on the pillars at their ends is calculated. Other Matlab scripts sum the pixel values of the pixels in a certain stress-fiber and compare it the to average pixel value in the whole frame to calculate the brightness of a stress-fiber.

4 Results

4.1 Force and strength

All fibroblast cells used in our experiments that were spread out on a pillar substrate were polarized, so the cells contain thick stress-fibers which are all oriented in the same direction. To see whether stress fibers are important in mediating forces in the cell we examined the direction of the orientation of the stress-fibers and the direction of the orientation of the forces that the cell exerts on the pillar substrate in 26 live but static cells. We averaged the direction of all stress-fibers found by a custom algorithm. Also the direction of all forces bigger than 3 nN in a frame were averaged. As shown in figure 13, the difference between the directions of the orientation of forces and stress-fibers in a cell is $2.3 \pm 21.4°$. This suggests that stress-fibers indeed are the principle mediator of forces that a cell exerts on a substrate.

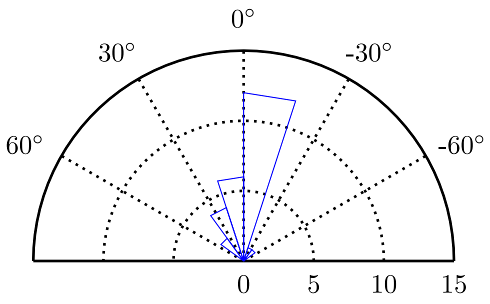


Figure 13: Histogram of the difference between the direction of the stress-fibers and the direction of the force on the pillars. The average is $2.3 \pm 21.4°$.

The main component of the cytoskeleton is actin. The small actin filaments as well as the stick stress-fibers are able to mediate forces. However, we showed that the thick stress-fibers are the main mediator of cell-to-substrate forces. This suggests a relation between fiber thickness and the force it exerts on the substrate. To test this, we look at the thickness of stress-fibers by assuming their brightness is a measure for their thickness. We compensate for differences between images due to exposure or fluorescence differences by looking at the brightness of a fiber relative to the brightness of the while image. However, comparison of the thicknesses of the stress-fibers to the forces that each stress-fiber exerts on the pillar substrate does not reveal a correlation between stress-fiber thickness and mediated force, as is shown in figure 14. This, however does not disprove the hypothesis that stress-fibers are important in mediating forces. It only shows that thick fibers do not necessarily mediate big forces although they probably can.

So, figure 13 suggests a relation between stress-fiber orientation and force exertion. However, figure 14 shows that, within in cell, thicker stress-fibers not necessarily do mediate big forces.

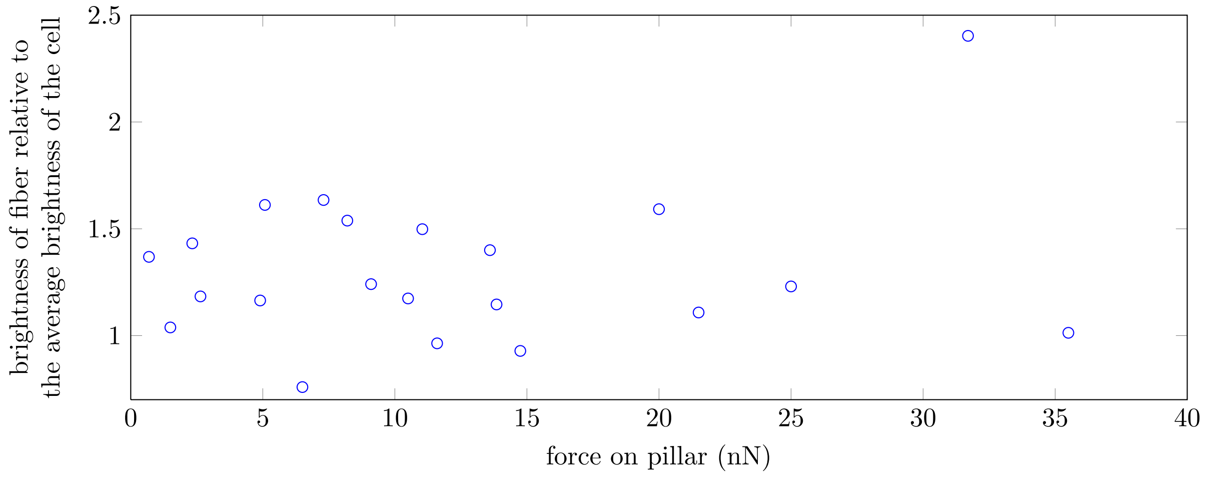


Figure 14: Forces on pillars and the brightness of the fiber attached to it. We assume the brightness of a fiber is proportional to its thickness. The brightness and the force appear to be uncorrelated.

4.2 Migrational forces

It has been suggested that cells use the leading edge of the cell called lamellipodium to move forward. To test if the leading edge of a moving cell exerts relatively big forces on the pillar substrate, we made time-lapse movies of moving cells. Three frames of such a time-lapse are shown in figure 15. Forces are then calculated from the deflections of the pillars. To track the progress of a cell over a pillar we divided each image into 4 μm diameter hexagonal areas using the grid for the calculated positions of non-deflected pillars. Then by plotting for each pillar the force versus time and the fraction of the area above the pillar that is taken up by the cell we show that indeed the leading edge of the cell exerts a relatively big force on the pillar substrate. Two examples of such plots are shown in figure 17.

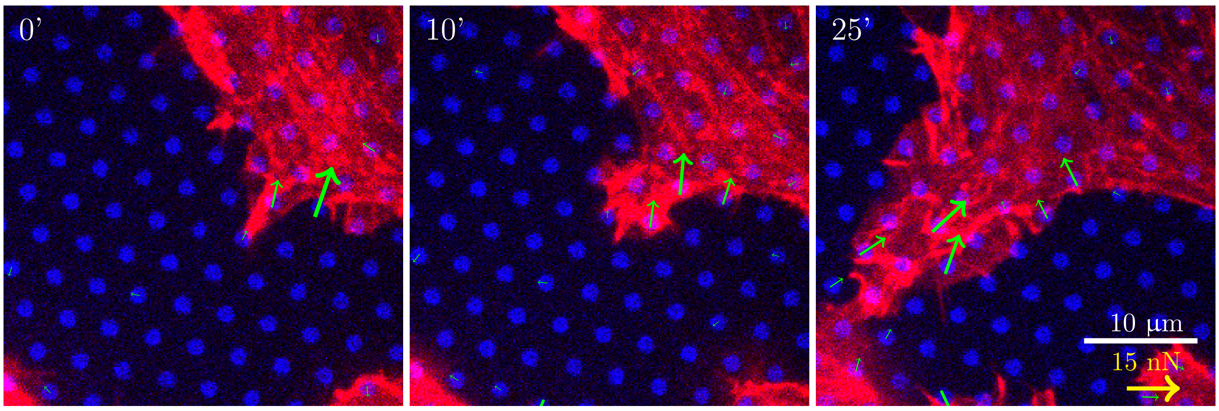


Figure 15: Timelapse of a moving cell, the length and direction of the arrows correspond to the magnitude and direction of the force. Actin is labelled red and stamped fibronectin is labelled blue. The force is mainly applied at the cell edge.

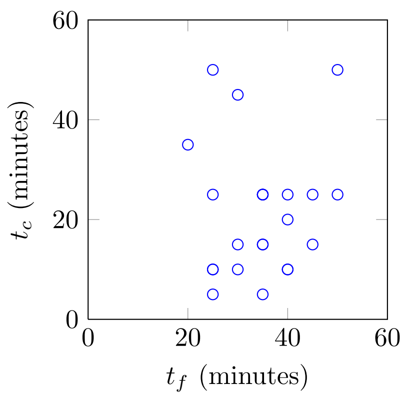


Figure 16: t_f plotted versus t_c.

As soon as a cell attaches to a pillar, it starts pulling on the pillar. The force exerted by the cell on a pillar is on average 6.2 ± 3.2 nN and lasts for (t_f) 35.5 ± 9.0 minutes. Over the 23 cells examined the time a cell needs to crawl over a pillar is (t_c) 23 ± 7.5 minutes. Their ratio ${}^{t_f}/{}_{t_c}$ is 2.3 ± 1.5 which can be used to calculate the size of the area that the cell uses to pull itself forward. Since we used 4 μm areas, the size of the area used for pulling is the area until 9.2 ± 6.0 μm from the leading edge. Although some theories predict that the part that pulls the cell forward is the lamellipodium, we do not call this part lamellipodium because from our experiments we cannot know how far the lamellipodium extends. Also, the plot of t_c versus t_f in figure 16 does not show any correlation between t_c and t_f. The reason might be that the size of the pulling part of the cell does vary hugely from cell to cell. Another thing that is likely to interfere is the fact that the cells we used are fibroblasts which are known to produce the ECM protein fibronectin. Fibronectin can connect multiple pillars together, distributing forces in the pillar array and thus obscuring measurements.

Although the leading edge of the cell is used to pull the cell forward, these forces are not the only forces present. In the rest of the cell there is friction and transient forces. These almost counteract the pulling force, resulting in only a slow crawling of the cell. The magnitude of these forces is generally below 2 nN as is shown in figure 15 where forces smaller than 2 nN are hidden.

Most cells exhibit the behavior explained above. However, about 10% of the cells exert only a small force of on average 2 nN when they move. Two examples are shown in figure 18. Where for most moving cells the force on a single pillar is 6.2 ± 3.2 nN and lasts for 7.1 ± 1.8 minutes, here there is no such big peak in the force. There still is a force, but it is spread over all pillars under the cell. This probably means that these cells move in a rolling-like motion as opposed to pulling themselves forward with a lamellipodium. We saw earlier that fibroblasts make fibronectin, so it might be that these cells do exert big forces, but that the pillars underneath these cells are interconnected with fibronectin and therefore are not deflected as much.

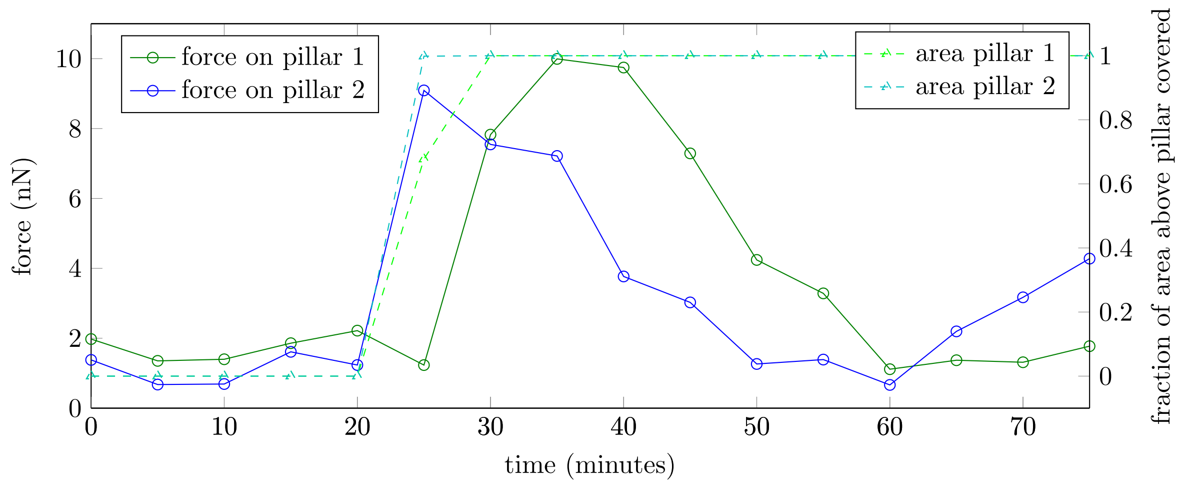


Figure 17: Forces on pillars and the extension of the cell above them. Some cells exert forces on pillars when they move over them.

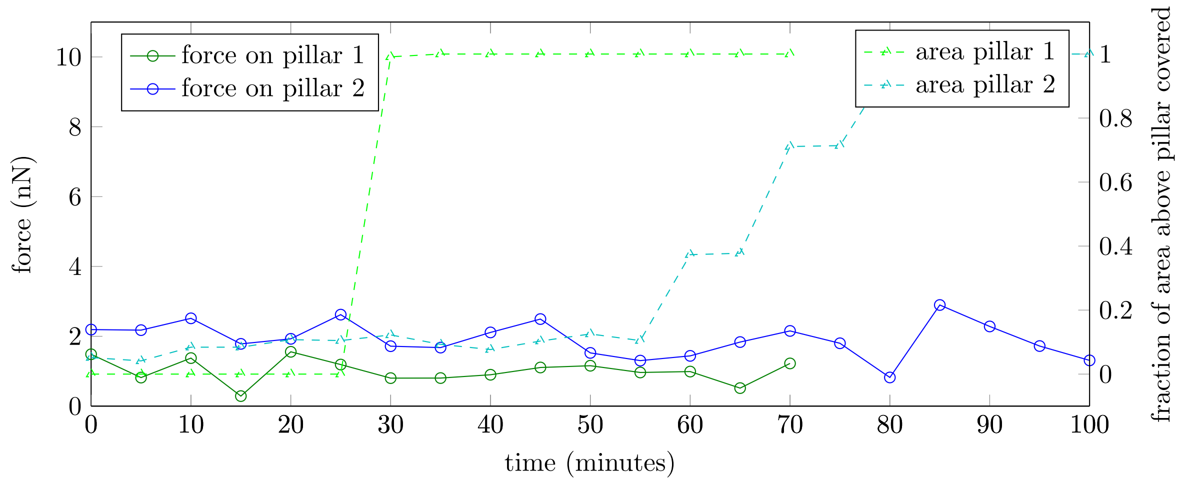


Figure 18: Forces on pillars and the extension of the cell above them. Some cells barely exert forces on pillars when they move over them.

4.3 Stress-fiber mediated active gel theory

Almost all cells imaged on pillars show curved edges with thick stress-fibers running along them. Only when cells were not completely spread, or dividing there were no curved edges with stress-fibers. We assume that the edges with a circular shape are only attached to the pillar substrate at the ends of the stress-fiber, the distance between the attachment points is the spanning distance d. Using the force on the pillars on the ends of the fibers and the radius R of the circular shape a surface tension σ can be found. Two examples are shown in figure 19. This suggests that the active gel theory is a good approximation to the behavior of the actin network in a cell.

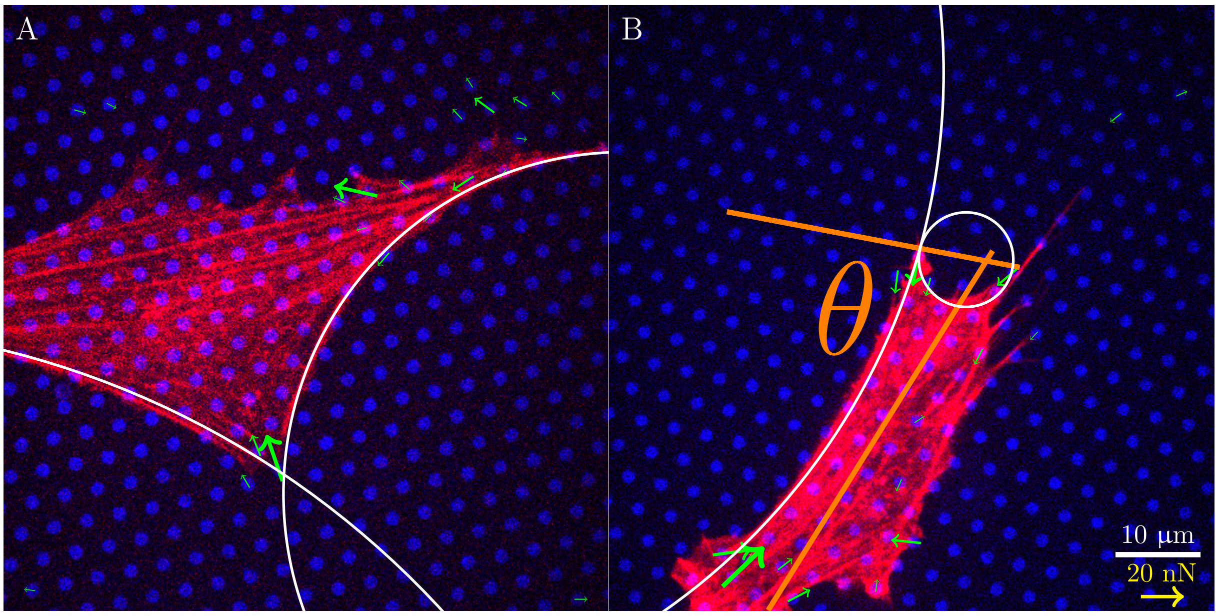


Figure 19: Live cells on pillars. Fibronectin stamped pillars in blue and mcherry labeled actin in red. Green arrows show forces bigger than 5 nN. **A.** Two edge stress fibers are fitted to a part of a circle. Surface tensions on the left and right stress fiber are 0.15 $^{\mathrm{nN}}/_{\mu\mathrm{m}}$ and 0.56 $^{\mathrm{nN}}/_{\mu\mathrm{m}}$ respectively. **B.** Tensions are 0.4 $^{\mathrm{nN}}/_{\mu\mathrm{m}}$ on the fiber fitted by the big arc, and 2.2 $^{\mathrm{nN}}/_{\mu\mathrm{m}}$ on the other fiber. The orange lines depict the general direction of the stress-fibers in the cell and the orientation of an edge of the cell. The angle between those is θ.

The tension-elasticity model described by equation 6 suggests a correlation between arc radius and spanning distance and line tension λ. However, we could not recover the suggested correlation between spanning distance, circle radius and line tension. Equation 6 is plotted in figure 20 with $AE = 500$ nN and $\alpha = 1$. The results from 51 arcs in different cells are also shown in figure 20B, but reveal no correlation. This suggests that when stress-fibers mediate the majority of the forces, the contribution due to the elasticity of actin is small.

Using equation 2 to calculate surface tensions gives surface tensions from less than 0.1 $^{\mathrm{nN}}/_{\mu\mathrm{m}}$ to more than 1.0 $^{\mathrm{nN}}/_{\mu\mathrm{m}}$. However, this surface tension does not just vary from cell to cell, but the surface-tension also differs between different stress-fibers in a single cell. Almost all the cells we imaged are highly polarized, and have have big basal stress-fibers running parallel in the cell. As is shown in figure 20A, edge stress-fibers tend to run parallel to the basal stress-fibers; in about 80% of the cases the angle between edge and basal stress-fibers is less than 30°.

As described in section 2.2, we expect the orientation of the edge stress-fibers relative to the basal stress-fibers to be important. In figure 21 the surface tension is plotted versus the sinus of this angle. This plot indeed suggests a relation between tension and orientation. By fitting a line described by equation 5 to this plot we can estimate the contribution of the active gel to the tension in an average 3T3-Fibroblast as $^{\lambda_{AG}}/_{R} = 0.08 \pm 0.07$ $^{\mathrm{nN}}/_{\mu\mathrm{m}}$, and the contribution of the basal stress-fibers as $^{\lambda_{SF}}/_{R} = 1.11 \pm 0.19$ $^{\mathrm{nN}}/_{\mu\mathrm{m}}$ or $^{\lambda_{SF}}/_{\lambda_{AG}} = 14 \pm 12$.

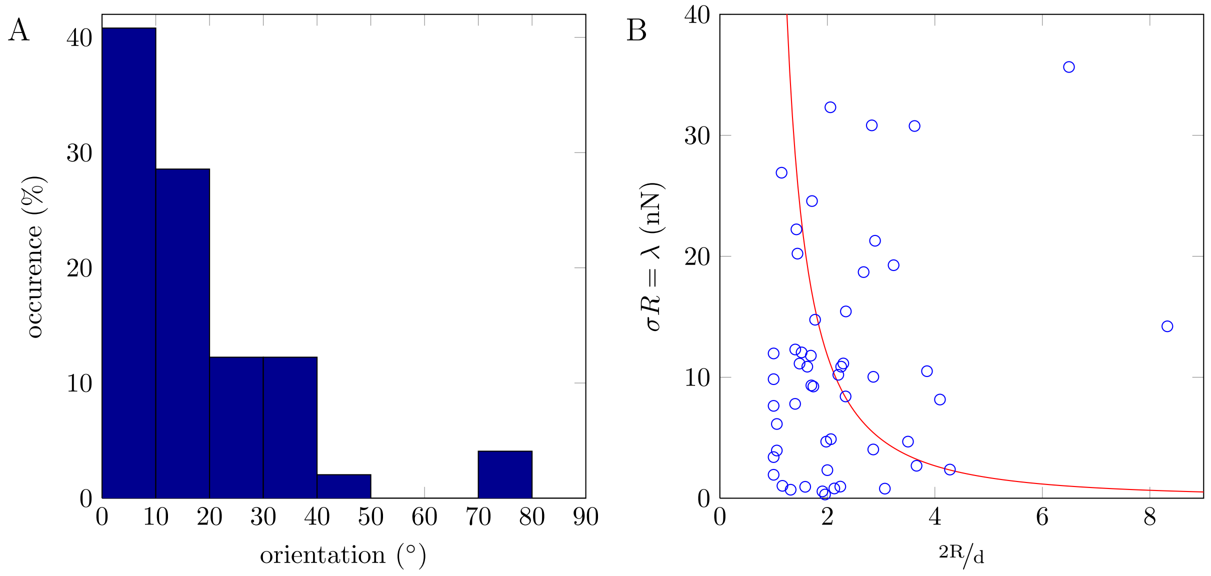


Figure 20: A. Orientation distribution of 49 edge stress-fibers relative to the basal stress-fibers. The edge stress-fibers tend to align parallel to the basal stress-fibers. **B.** Red: equation 6 with $AE = 500$ nN and $\alpha = 1$. Blue: data from 51 circular edges, no correlation was found between $^{R}/_{d}$ and λ.

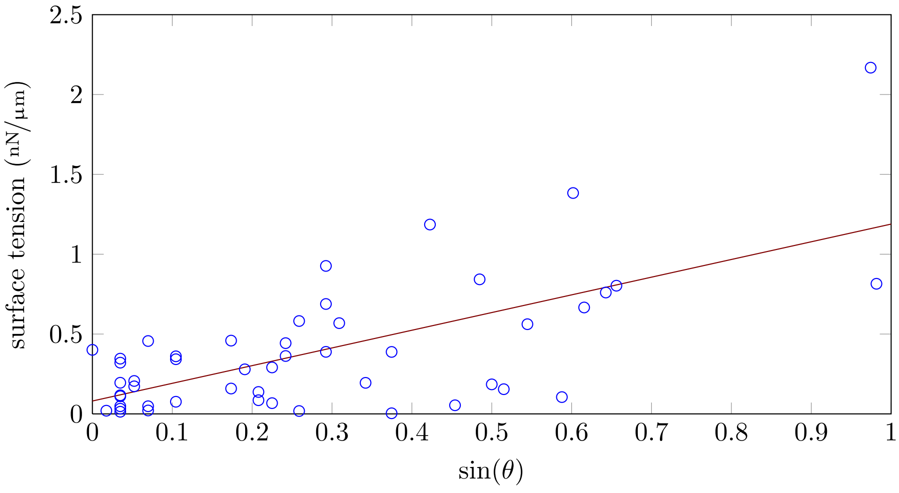


Figure 21: Surface tension ($^{\mathrm{nN}}/_{\mu\mathrm{m}}$) versus orientation. Data points are blue. A fit $\sigma = 0.08 + 1.11 \cdot \sin\theta$ in red shows the contributions of the active gel and basal stress-fibers to the surface tension in an average cell.

5 Discussion

Our experiments involve fibroblast cells on a pillar array coated with fibronectin. Fibroblasts are known to produce fibronectin. In this way the cells are able to interconnect pillars. These connections can distribute the forces that a cell exerts over different pillars. So it is possible that on some pillars that are not deflected much still a big force is exerted that is then distributed over several pillars. However, we mainly look at big forces, and in those cases the majority of the force is generated by the cell on the pillar where the force is measured. The agreement of our result with theory further confirms that although fibronectin may be interconnecting pillars, a pillar array is still a good method to measure forces that cells exert on a substrate.

Our analysis was done on two-dimensional pictures of the bases of cells, so we do not know how the actin network outside the basal plane affects our results. However, the active gel theory works in either two or three dimensions. Also our extension to the active gel theory is valid in both cases. Our extension is an approximation the interaction between stress-fibers and tension in the cell, however, any more detailed model will require experiments that control stress-fiber growth and deliver results with better accuracy. Despite our models simplicity, our results suggest that the stress-fiber mediated active gel theory describes the interaction between stress-fibers and tension in the cell.

6 Conclusion

Using confocal microscopy and pillar arrays we showed that in accordance with recent theories [8], cells mainly exert forces with an average magnitude of 6.2 ± 3.2 nN on a substrate via the leading edge of the cell, which extends 9.2 ± 6.0 µm the cell. We were also able to show that in a polarized cell stress-fibers are the main mediator for forces by showing a correlation between the directions of force and basal stress-fiber orientation. The average difference between those directions was $2.3 \pm 21.4°$ Furthermore, we show that there is no correlation between stress-fiber thickness and the force that that stress-fiber transmits.

Our experiments show a correlation between orientation of fibers relative to the edge of the cell and the surface tension. Our experiments suggest a contribution to the contractility of a cell from stress-fibers up to 14 ± 12 times higher than the contribution from the active gel, depending on the relative orientation of basal stress-fibers and the edge of the cell. Therefore we propose an extension to the active gel theory [13, 14] incorporating basal stress-fibers. We show that our results cannot be explained by the tension-elasticity model, because the contribution of stress-fibers to tension in fibroblasts is bigger than the correction due to the elasticity of the actin network. However, it might be possible to combine the tension-elasticity and stress-fiber mediated active gel theory into one model.

7 References

[1] Adam J Engler, Shamik Sen, H Lee Sweeney, and Dennis E Discher. Matrix elasticity directs stem cell lineage specification. *Cell*, 126(4):677–89, August 2006.

[2] A Zemel, F Rehfeldt, A E X Brown, D E Discher, and S A Safran. Optimal matrix rigidity for stress fiber polarization in stem cells. *Nature physics*, 6(6):468–473, June 2010.

[3] Tubulin, actin and DNA distribution in 3T3 cells. `http://www.olympusmicro.com/galleries/confocal/cells/3t3/3t3sb3.html`.

[4] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Molecular biology of the cell.* Garland Science, 5th edition, 2008.

[5] Irina Kaverina, Olga Krylyshkina, and J Victor Small. Regulation of substrate adhesion dynamics during cell motility. *The international journal of biochemistry & cell biology*, 34(7):746–61, July 2002.

[6] Jianping Fu, Yang-Kao Wang, Michael T Yang, Ravi A Desai, Xiang Yu, Zhijun Liu, and Christopher S Chen. Mechanical regulation of cell function with geometrically modulated elastomeric substrates. *Nature methods*, 7(9):733–6, September 2010.

[7] Michael T Yang, Jianping Fu, Yang-Kao Wang, Ravi A Desai, and Christopher S Chen. Assaying stem cell mechanobiology on microfabricated elastomeric substrates with geometrically modulated rigidity. *Nature protocols*, 6(2):187–213, February 2011.

[8] Ulrich S Schwarz and Margaret L Gardel. United we stand - integrating the actin cytoskeleton and cell-matrix adhesions in cellular mechanotransduction. *Journal of cell science*, 125(July):3051–3060, July 2012.

[9] F Julicher, K Kruse, J Prost, and J Joanny. Active behavior of the Cytoskeleton. *Physics Reports*, 449(1-3):3–28, September 2007.

[10] Arisa Uemura, Thuc-Nghi Nguyen, Amanda N Steele, and Soichiro Yamada. The LIM domain of zyxin is sufficient for force-induced accumulation of zyxin during cell migration. *Biophysical journal*, 101(5):1069–75, September 2011.

[11] Christopher S Chen. Mechanotransduction - a field pulling together? *Journal of cell science*, 121(Pt 20):3285–92, October 2008.

[12] Cyrus A Wilson, Mark A Tsuchida, Greg M Allen, Erin L Barnhart, Kathryn T Applegate, Patricia T Yam, Lin Ji, Kinneret Keren, Gaudenz Danuser, and Julie A Theriot. Myosin II contributes to cell-scale actin network treadmilling through network disassembly. *Nature*, 465(7296):373–7, May 2010.

[13] Ilka B Bischofs, Franziska Klein, Dirk Lehnert, Martin Bastmeyer, and Ulrich S Schwarz. Filamentous network mechanics and active contractility determine cell and tissue shape. *Biophysical journal*, 95(7):3488–96, October 2008.

[14] Ilka Bischofs, Sebastian Schmidt, and Ulrich Schwarz. Effect of Adhesion Geometry and Rigidity on Cellular Force Distributions. *Physical Review Letters*, 103(4):1–4, July 2009.

8 Acknowledgments

I would like to thank Hedde van Hoorn and Thomas Schmidt for their supervision on this work. It was nice to be a part of the mechanosensing project. Furthermore, thanks to all members of the Physics of Live Processes group for the wonderfull time. I´m happy to be privileged to continue to work in this symbiotic environment. Let´s not forget the homefront, without your support and interest this thesis wouldn´t have been possible. However, above all: Soli Deo Gloria.

Figure 22: Thomas Schmidt, Wim Pomp and Hedde van Hoorn

9 Appendix

9.1 Stress-fiber analysis

The following scripts are written in Matlab to find stress-fibers in images. A general description is given in section 3.4.2.

sffind.m

```
function [SF,SD,BP,EP,GSF,branches,E,B] = sffind(im)
% usage: [SF,SD,BP,EP,GSF,branches,E,B] = sffind(im)
% ------------------------------------------------------------------------
% tries to find stress fibers
%
% input:    im: 2D grayscale image
% output:   SF: BW image with stress fibers (logical image)
%           SD:               skeleton (logical image)
%           BP:               branchpoints (logical image)
%           EP:               endpoints (logical image)
%           GSF: image in which branches are numbered (uint16 image)
%           branches: branches info struct(i)
%               i: number of the branch, corresponding to GSF
%               Area: area of the branch (pixels)
%               Orientation: orientaion of the branch (    )
%               PixelIdxList: linear list of pixel indices in the original
%               Endpoints: number of the endpoint(s) to this branch
%               Branchpoints: number of the branchpoint(s) to this branch
%           E/B: endpoints/branchpoints info struct(i) containing Area and
%                   PixelIdxList
%               i: number of the ep/bp, corresponding to
%                   branches.Endpoints/Branchpoints
%               AdjBranch: number of the branch(es) adjacent to the ep/bp
% ------------------------------------------------------------------------

if nargin<1,help sffind;return;end

im=double(im);

B=LowpassImage(im,30);
BI=(im-B);
thr=std(BI(:));
DBW=imfill(LowpassImage(BI,2)>thr,'holes');
M=imdilate(bwmorph(bwmorph(DBW,'shrink',3),'clean'),strel('disk',1));
F=[regionprops(M,'Area','PixelIdxList')];

%find all features with a big size
SF=zeros(size(im));
for i=1:length(F)
    if(F(i).Area>50)
        SF(F(i).PixelIdxList)=1;
    end
end

%make skeleton image and find branch and endpoints
skel=bwmorph(SF,'skel',12);
thin=bwmorph(skel,'thin');
BP=bwmorph(thin,'branchpoints');
EP=bwmorph(thin,'endpoints');
SD=bwmorph(thin,'diag');

%make sure branches are cut at branchpoints
BP = expbp(SD,BP);

%give each branch a number
NBS=numberbranches(SD,BP,EP);

F=[regionprops(NBS,'Area','PixelIdxList','Orientation')];

%treat small pieces as branchpoints
NBSG=zeros(size(NBS));
for i=1:length(F)
    if F(i).Area<=10
        BP(F(i).PixelIdxList)=1;
    else
        NBSG(F(i).PixelIdxList)=i;
    end
end

%identify adjoining fibers with the same orientation
try
    GSF=gluefibers(NBSG,BP,F);
catch err
    cprintf([1,0.5,0],err.message);
    GSF=zeros(size(NBSG));
end
```

```
F=[regionprops(int32(GSF),'Area','PixelIdxList','Orientation')];

%reorder
GSF=zeros(size(im));
j=1;
for i=1:length(F);
    if F(i).Area
        branches(j)=F(i);
        GSF(F(i).PixelIdxList)=j;
        j=j+1;
    end
end

%the image has changed, so we need to determine branch- and endpoints again
[E,B]=getendpoints(GSF);

%list endpoints in branches struct
for i=1:length(E)
    for j=1:length(E(i).AdjBranch)
        if ~isfield(branches(E(i).AdjBranch),'Endpoints')
            branches(E(i).AdjBranch(j)).Endpoints=i;
        else
            branches(E(i).AdjBranch(j)).Endpoints=[branches(E(i).AdjBranch(j)).Endpoints,i];
        end
    end
end

%list branchpoints in branches struct
for i=1:length(B)
    for j=1:length(B(i).AdjBranch)
        if ~isfield(branches(B(i).AdjBranch),'Branchpoints')
            branches(B(i).AdjBranch(j)).Branchpoints=i;
        else
            branches(B(i).AdjBranch(j)).Branchpoints=[branches(B(i).AdjBranch(j)).Branchpoints,i];
        end
    end
end

GSF=uint16(GSF);
SF=logical(SF);
if exist('branches','var')
    branches=transpose(branches);
else
    branches=struct;
    branches.PixelIdxList=[];
    branches.Orientation=[];
    branches.Endpoints=[];
    branches.Branchpoints=[];
end
end
```

expbp.m

```
function BP = expbp(SD,BP)
% usage: BP = expbp(SD,BP)
%
% Finds true branchpoints
%
% SD: skeleton image
% BP: branchpoints image

s=size(SD);
t=s(1);
u=s(2);

idx=[-t,-1,t,1];
idy=[-t+1,-t-1,t-1,t+1,-t+1];

bp=find(BP);

for i=1:length(bp)
    for j=1:length(idx)
        li=bp(i)+idx(j);
        lj=bp(i)+idy(j);
        lk=bp(i)+idy(j+1);
        if(SD(li)&&SD(lj)&&SD(lk))
            BP(li)=1;
        end
    end
end
end
```

numberbranches.m

```matlab
function [NBS,branches,bp] = numberbranches(SD,BP,EP)
% usage: [NBS,branches,bp] = numberbranches(SD,BP,EP)
%
% Gives each branch a number
%
% SD: skeleton image
% BP: branchpoints
% EP: endpoints
%
% NBS:      image with numbered branches
% branches: struct with properties of branches
% bp:       list of branchpoints

s=size(SD);
t=s(1);
u=s(2);
NBS=BP;
bp=cat(1,find(BP),find(EP));
SD=SD-BP-EP;

idx=[-t-1,-t,-t+1,-1,1,t-1,t,t+1];

branches=struct;
branches(1).Endpoints=[];
branches(1).PixelIdxList=[];

k=2;
for i=1:length(bp)
    for j=1:length(idx)
        li=bp(i)+idx(j);
        if(li>0&&li<=t*u)
            if(SD(li)==1)
                %disp(['i=',num2str(i),' li=',num2str(li)]);
                B=(imfill(~SD,li)-(~SD))*k;
                NBS=NBS+B;
                SD=SD+B;
                [y,x]=ind2sub(s,bp(i));
                branches(k).Endpoints=[x,y];
                branches(k).PixelIdxList=find(B);
                k=k+1;
            elseif(SD(li)>1)
                [y,x]=ind2sub(s,bp(i));
                if ~isempty(branches(SD(li)-1).Endpoints)
                    if branches(SD(li)-1).Endpoints(end,:)~=[x,y]
                        branches(SD(li)-1).Endpoints=cat(1,branches(SD(li)-1).Endpoints,[x,y]);
                    end
                end
            end
        end
    end
end
end
```

gluefibers.m

```matlab
function GSF = gluefibers(NBS,BP,F)
% usage: GSF = gluefibers(NBS,BP,F)
%
% This function glues two fibers at every bp, it glues the two with the
% best matching orientations
%
% NBS: image with numbered branches
% BP:  image with branchpoints
% F:   struct with properties of branches
%
% GSF: new image with glued and numbered stress-fiber branches

s=size(NBS);
t=s(1);
u=s(2);

GSF=NBS;
B=[regionprops(BP,'Area','PixelIdxList')];

for i=1:length(B)
    %make a list of pixels bordering a bp
    idx{i}=border(BP,B(i).PixelIdxList);

    orientations=nan(length(idx{i}),1);
    num=nan(length(idx{i}),1);
    o=nan(length(idx{i}));
    ff=nan(length(idx{i}),1);

    %make a list of orientations of branches around the bp, each branch
    %occurs only once!
    for j=1:length(idx{i})
```

```matlab
        if(idx{i}(j)>0&&idx{i}(j)<=t*u&&NBS(idx{i}(j))>1&&nanmin(abs(num-NBS(idx{i}(j))))~=0)
            num(j)=NBS(idx{i}(j));
            orientations(j)=F(NBS(idx{i}(j))).Orientation;
        end
    end

    %relate orientations of branches in o
    for q=1:length(orientations)
        for r=q:length(orientations)
            if (orientations(r)~=0&&orientations(q)~=0&&r~=q)
                o(r,q)=abs(orientations(r)-orientations(q));
                if o(r,q)>90,o(r,q)=abs(o(r,q)-180);end
            end
        end
    end

    %find connections
    [m,id]=nanmin(o(:));
    while ~isnan(m)
        [x,y]=ind2sub(size(o,1),id);
        ff(y)=x;
        o(x,:)=NaN;
        o(:,x)=NaN;
        o(y,:)=NaN;
        o(:,y)=NaN;
        [m,id]=nanmin(o(:));
    end

    %sort to make sure bottomright branches always get the number from
    %topleft
    for j=1:length(ff)
        if ff(j)<j
            ff(ff(j))=j;
            ff(j)=NaN;
        end
    end

    %glue
    for j=1:length(idx{i})
        if(idx{i}(j)>0&&idx{i}(j)<=t*u&&~isnan(ff(j)))
            px=F(NBS(idx{i}(ff(j)))).PixelIdxList;
            GSF(px)=GSF(idx{i}(j));

            F( GSF(idx{i}(j)) ).PixelIdxList = cat(1,F( GSF(idx{i}(j)) ).PixelIdxList,F( GSF(idx{i}(ff
                (j))) ).PixelIdxList);

        end
    end
end

%give the BPs a color
for i=1:length(B)
    A=GSF(idx{i});
    A(A==0)=[];
    GSF(B(i).PixelIdxList)=mode(A);
end
end
```

getendpoints.m

```matlab
function [E,B] = getendpoints(GSF)
% usage: [E,B] = getendpoints(GSF)
%
% Finds endpoints end branchpoints to branches
%
% GSF: numbered stress-fiber image
%
% E: struct with endpoints
% B: struct with branchpoints

s=size(GSF);
t=s(1);
u=s(2);

EP=bwmorph(bwmorph(GSF>1,'thin'),'endpoints');
BP=bwmorph(bwmorph(GSF>1,'thin'),'branchpoints');

E=[regionprops(EP,'Area','PixelIdxList')];
B=[regionprops(BP,'Area','PixelIdxList')];

for i=1:length(E)
    %make a list of pixels bordering an ep
    idx{i}=border(EP,E(i).PixelIdxList);
    A=GSF(idx{i});
    A(A==0)=[];
    E(i).AdjBranch=unique(A);
```

```
end

for i=1:length(B)
    %make a list of pixels bordering a bp
    idx{i}=border(BP,B(i).PixelIdxList);

    A=GSF(idx{i});
    A(A==0)=[];
    B(i).AdjBranch=unique(A);
end

%remove false branchpoints
for i=1:length(B),l(i)=length(B(i).AdjBranch);end
if exist('l'),B(l<2)='';end

end
```

9.2 Horizontal drift correction

The following script uses Fourier transformations to determine the offset of a frame relative to the first frame.

corrfft.m

```
function [d,cfunc] = corrfft(imageb1,imageb2,imager1,imager2,pd)
% usage: [d,cfunc] = corrfft(pillarimage(1),pillarimage(n),image(n-1),image(n),d(n-1));
%
% uses fft's to calculate the correlation function between images and thus
% determine the offset relative to the first frame
%
% input:
%   pillarimage(1): first frame from the pillardata
%   pillarimage(n): current frame from the pillardata
%   image(n-1):     previous frame from 'other' data
%   image(n):       current frame from 'other' data
%   d(n-1):         offset of the previous frame
%
% output:
%   d:              offset (x,y)
%   cfunc:          correlation function

if nargin<5,help corrfft;return;end

%first shift the n-1 frame so that its offset is 0
imager1=shiftimage(imager1,pd);

cfunc=ifft2(fft2(imageb1).*conj(fft2(imageb2))).*ifft2(fft2(imager1).*conj(fft2(imager2)));

[~,i]=max(cfunc(:));
[x,y]=ind2sub(size(cfunc),i);

d=[x-1,y-1];

%peak at x=511 means xoffset=-1 etc
for k=1:2,if d(k)>256,d(k)=d(k)-512;end;end

end
```

9.3 Pillar identification

The following scripts identify pillars in consecutive frames

connectpillars.m

```
function [defl,tempdefl] = connectpillars(defl,tempdefl,shift)
% usage: [defl,tempdefl] = connectpillars(defl,tempdefl,shift);
% ---------------------------------------------------------------------------
% Reorders pillars in defl so that in each frame pillar n alsways holds
% defl.xy(n,:), defl.xyref(n,:), etc
%
% defl=connectpillars(defl,shift)
%
% defl: struct with where within each sub the last array index is the
% framenumber
% shift: xy-correction as the output from alloffsets

if nargin<1,help connectpillars;return;end
if nargin==3,defl=shiftdefl(defl,shift);end

%%initiate the coordinates against which pillars will be sought
%tempdefl.xyref=defl{1}.xyref;
```

```matlab
%h = waitbar(0,);
cpb=create_bar('Time','on','Text',sprintf('Sorting pillars, frame 1 of %d, total %d pillars.',size(
    defl,2),size(tempdefl.xyref,1)),'Length',10);

for i=2:size(defl,2)
    if (~isempty(defl{i})&&~isempty(fieldnames(defl{i})))
        %find pillars in a frame and sort them
        c=pillarfind(tempdefl,defl{i});
        [defl{i},extra]=pillarsort(defl{i},c);

        %add extra pillars to tempdefl
        s=length(tempdefl.xyref);
        if size(extra.xyref,1)
            for j=1:size(extra.xyref,1),
                tempdefl.xyref(s+j,:)=extra.xyref(j,:);
            end
        end
        %waitbar(i/size(defl,2),h,sprintf('Sorting pillars, frame %d of %d, total %d pillars.',i,size(
            defl,2),size(tempdefl.xyref,1)));
        cpb.setTextandValue(i/size(defl,2),sprintf('Sorting, frame %d/%d, %d pillars.',i,size(defl,2),
            size(tempdefl.xyref,1)));
    end
end

s=size(tempdefl.xyref,1);
defl=deflstruct2arr(defl,s);

%delete(h);
cpb.stop;
scatter(tempdefl.xyref(:,1),tempdefl.xyref(:,2));

%check for duplicates
[~,d]=pillarfind(tempdefl,tempdefl);
x=(sum(d(:))-size(d,1))/2;
if x
    %[~,c]=find(d-eye(size(d,1)));
    cprintf([1,0.3,0],['Warning: ',num2str(x),' extra pillars found.\n']);
    %for i=1:size(r),disp([num2str(r(i)),'=',num2str(c(i))]);end
end
end
```

pillarfind.m

```matlab
function [c,d] = pillarfind(defl1,defl2)
% usage: [c,d] = pillarfind(defl1,defl2)
% ------------------------------------------------------------------------
% correlates pillars in two frames
% inputs:
%   defl1, defl2: deflection structs, xy-correction needs to be done first!
%
% output:
%   c:  c(i)=j; i in defl1 corresponds to j in defl2

if nargin<2,help pillarfind;return;end

for i=1:size(defl1.xyref,1)
    for j=1:size(defl2.xyref,1)
        d(i,j)=norm(defl1.xyref(i,:)-defl2.xyref(j,:))<10;
    end
    [v,j]=max(d(i,:));
    c(i)=v*j;
end
end
```

pillarsort.m

```matlab
function [deflcorr,extra] = pillarsort(defl,c)
% usage: [deflcorr,extra] = pillarsort(defl,c)
% ------------------------------------------------------------------------
% Sorts deflections according to c and puts leftover deflections in extra
%
% inputs:
%       defl:       deflection structure
%       c:          array c(i)=j, pillar j gets place i
%
% outputs:
%       deflcorr:   deflection structure, sorted
%       extra:      defl structure (only xyref) of pillars not defined in c

if nargin<2,help pillarsort;return;end

done=[];
for i=1:length(c)
    if c(i)
        deflcorr.xy(i,:)=defl.xy(c(i),:);
        deflcorr.xyref(i,:)=defl.xyref(c(i),:);
        deflcorr.xydefl(i,:)=defl.xydefl(c(i),:);
        deflcorr.absdefl(i,1)=defl.absdefl(c(i),1);
        done(c(i))=1;
    else
        deflcorr.xy(i,:)=[NaN,NaN];
        deflcorr.xyref(i,:)=[NaN,NaN];
        deflcorr.xydefl(i,:)=[NaN,NaN];
        deflcorr.absdefl(i,1)=NaN;
    end
end

if length(done)-sum(done)
    s=size(defl.xyref,1);
    for i=1:length(done)-sum(done)
        [~,j]=min(done);
        extra.xyref(i,:)=defl.xyref(j,:);
        deflcorr.xy(s+i,:)=defl.xy(j,:);
        deflcorr.xyref(s+i,:)=defl.xyref(j,:);
        deflcorr.xydefl(s+i,:)=defl.xydefl(j,:);
        deflcorr.absdefl(s+i,1)=defl.absdefl(j,1);
        done(j)=1;
    end
else
    extra.xyref=[];
end
end
```

9.4 Circle fitting

Fitting circles to identified stress-fiber images is done using the following Matlab script:

curvature.m

```
function [r,c,curve]=curvature(GSF,n)
% usage: [r,c,curve]=curvature(GSF,n)
%
% Fits a circle to stress-fibers
%
% GSF:   2D image with stress-fibers, as generated by sffind
% n:     index or indices of the stress-fiber that should be fitted
%
% r:     radius of the fitted circle
% c:     center of the fitted circle
% curve: linear indices of the stress-fiber

if nargin<2,help curvature;return;end
if length(size(GSF))~=2,help curvature;return;end

curve=zeros(size(GSF));
for i=1:length(n)
    curve=(GSF(:,:,1)==n(i))+curve;
end
[x,y]=find(curve);
mx = mean(x); my = mean(y);
X = x - mx; Y = y - my; % Get differences from means
dx2 = mean(X.^2); dy2 = mean(Y.^2); % Get variances
t = [X,Y]\(X.^2-dx2+Y.^2-dy2)/2; % Solve least mean squares problem
a0 = t(1); b0 = t(2); % t is the 2 x 1 solution array [a0;b0]
r = sqrt(dx2+dy2+a0^2+b0^2); % Calculate the radius
a = a0 + mx; b = b0 + my; % Locate the circle's center
%curv = 1/r; % Get the curvature
c=[b,a];
end
```

9.5 General Control

General Control controls the autofocus hardware and is written in NI Labview 2011. Because it is written in a graphical programming language, including all the code with all possible cases is almost impossible. Therefore, we only include the block diagram of General Control itself in figure 23. The program is divided into several parallel loops that each control a different piece of hardware or a part of the user interface. The grey blocks in figure 23 are the loops that contain the code for the user interface. The loops that control the hardware is in subfunctions which in figure 23 are displayed as green squares near the top of the image. Each loop can communicate variables to another loop by making use of ´handles´, subfunctions which can store variables and do small operations on them.

All hardware but the Marzhauser microscope stage could be controlled using Labview. Therefore we made a Python script that executes functions from a dynamic link library provided by Marzhauser. The

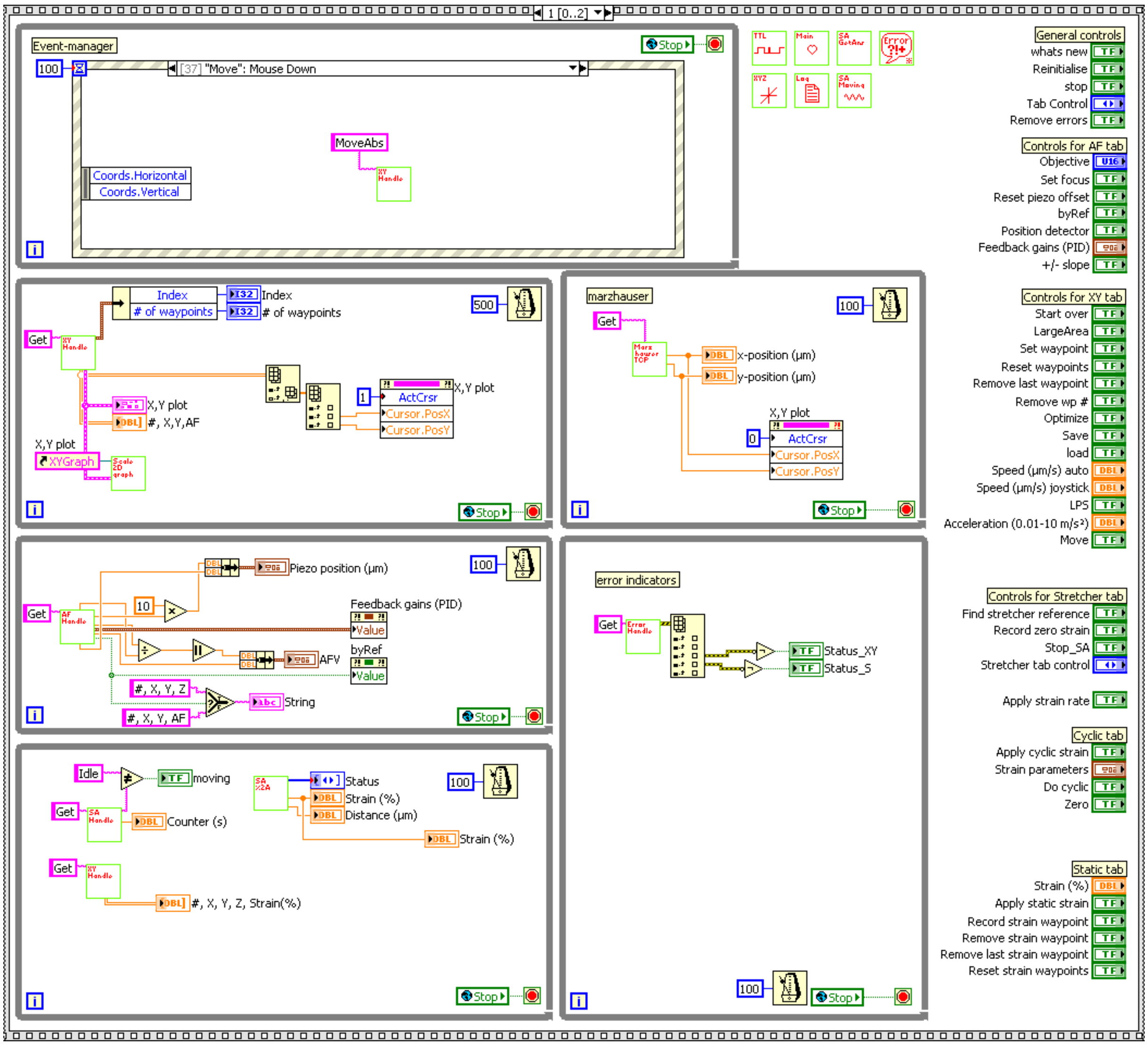


Figure 23: General Control block diagram, code is written in NI Labview 2011

script can communicate with General Control via the TCP/IP protocol.

mh–tcp.py

```
#!/usr/bin/env python
# coding: utf-8
# -----------------------------------------------------------------------
# Marzhauzer xy-stepper python script, controllabe by a string over tcp:
#
# -a acceleration: set acceleration (m/s^2)
# -j: turns joystick on
# -l: logging on for this execution
# -m -x xpos -y ypos, move to [xpos, ypos] (um) optional: -v for speed
#    just this move
# -v speed: set speed um/s)
#
# Wim Pomp BSc.
# -----------------------------------------------------------------------

#use ctypes to call C-library
from ctypes import *
from optparse import OptionParser
import sys
import socket
import re

TCP_IP = '127.0.0.1'
TCP_PORT = 5005
BUFFER_SIZE = 64  # Normally 1024, but we want fast response

```

```python
parser = OptionParser()
parser.add_option('-l', action='store_true', dest='log', default=False)
(opts, args) = parser.parse_args()

log   = opts.log

#constants
char  = c_char()
dnul  = c_double(0)
deen  = c_double(1)

#load Marzhauser driver
mh = windll.LoadLibrary("E:\\Programs\\Marzhauser\\lstep4x.dll")

#error checking function
def chk(err):
  if err:
    raise SystemExit('Error:', err)

#decode message
def options(message):
  p = OptionParser()
  p.add_option('-a', type='float', dest='acc', default=0)
  p.add_option('-v', type='float', dest='speed', default=0)
  p.add_option('-x', type='float', dest='xposs', default=0)
  p.add_option('-y', type='float', dest='yposs', default=0)
  p.add_option('-j', action='store_true', dest='joy', default=False)
  p.add_option('-m', action='store_true', dest='move', default=False)
  p.add_option('-q', action='store_true', dest='stop', default=False)
  #can't communicate empty strings, use -p to get pos only
  p.add_option('-p', action='store_true', dest='getpos', default=True)
  arguments=re.split(r"\s",message)
  (o, args) = p.parse_args(arguments)
  return o

#actually do things with the stage
def doit(op):
  #variables from options
  xposs = c_double(op.xposs/1000)
  yposs = c_double(op.yposs/1000)
  acc   = c_double(op.acc)
  speed = c_double(op.speed/1000)

  #boolians from options
  move  = op.move
  joy   = op.joy

  #byref variables
  xpos  = c_double(0)
  ypos  = c_double(0)
  zpos  = c_double(0)
  apos  = c_double(0)
  xspeed = c_double(0)
  yspeed = c_double(0)
  zspeed = c_double(0)
  aspeed = c_double(0)
  jo    = c_bool(False)
  man   = c_bool(False)
  pc    = c_bool(False)
  enc   = c_bool(False)

  #The labview libary is the best source on how to call a certain function, LSTEP*.pdf=BS
  mh.LSX_GetVel(1,byref(xspeed),byref(yspeed),byref(zspeed),byref(aspeed))
  if acc: #set acceleration
    mh.LSX_SetAccel(1,acc,acc,dnul,dnul)
  print "Acceleration set to "+str(op.acc)+" m/s^2."
  if speed: #set speed
    mh.LSX_SetVel(1,speed,speed,deen,deen)
  if not move:
    print "Speed set to "+str(op.speed)+" um/s."
  if move: #move, first get joystick (js) status settings and turn js of
    mh.LSX_GetJoystick(1,byref(jo),byref(man),byref(pc),byref(enc))
    mh.LSX_SetJoystickOff(1)
  print "Moving... speed: "+str(op.speed)+" um/s."
    mh.LSX_MoveAbs(1, xposs, yposs, dnul, dnul, 1) #<-- actual movement command
  print "Moved to: ("+str(op.xposs)+"; "+str(op.yposs)+") um."
    if jo: #afterwards make js on if it was on before
      mh.LSX_SetJoystickOn(1,True,False)
    if speed: #reset speed
      mh.LSX_SetVel(1,xspeed,yspeed,zspeed,aspeed)
  if joy: #joystick on
    mh.LSX_SetJoystickOn(1,True,False)
  print "Joystick on"
  mh.LSX_GetPos(1,byref(xpos),byref(ypos),byref(zpos),byref(apos))
  xpos = 1000*float((str(xpos)[9:-1]+"00000000000")[0:10])
  ypos = 1000*float((str(ypos)[9:-1]+"00000000000")[0:10])
  zpos = 1000*float((str(zpos)[9:-1]+"00000000000")[0:10])
  apos = 1000*float((str(apos)[9:-1]+"00000000000")[0:10])
  pos=[xpos,ypos,zpos,apos]
  return pos

#bind to socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((TCP_IP, TCP_PORT))
s.listen(1)
print "I am a python script that talks to the marzhauser stage. Please don't kill me, because if you
    do, the stage will stop working, and you will have to restart GC.\n"
print "Bound to "+str(TCP_IP)+":"+str(TCP_PORT)+"."

#connect to stage
chk(mh.LSX_ConnectSimple(1, 4, char, 0, log))
chk(mh.LSX_SetActiveAxes(1,3))

while 1:
  #connect to socket
  (conn, addr) = s.accept()
  message = conn.recv(BUFFER_SIZE)
  op=options(message)
  if op.stop: break
  pos=doit(op)
  conn.send(str(pos[0])+";"+str(pos[1])+"\r\n")
  conn.close()

#disconnect
conn.send("stopped\r\n")
conn.close()
print "Stopped."
chk(mh.LSX_Disconnect(1))
```