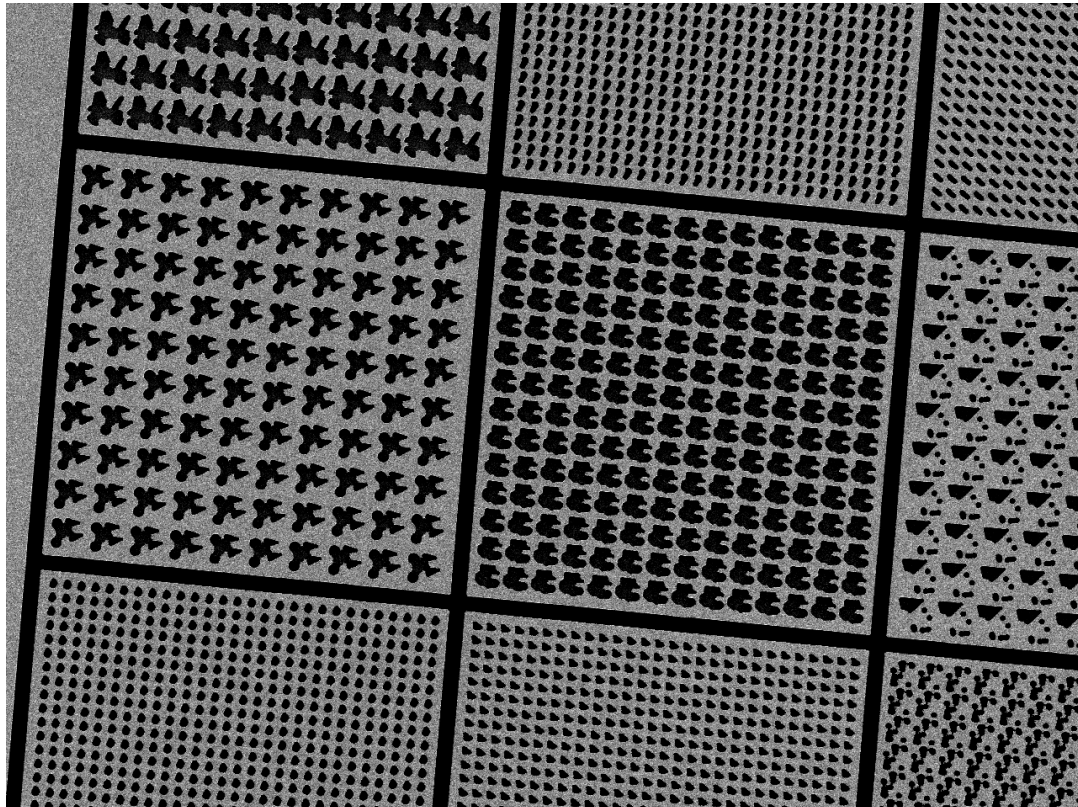


Algorithmic feature generation for microscale topographies

Kamiel Cornelissen

Master of Science thesis
Applied Mathematics

June 2008 - May 2009



Supervisors Discrete Mathematics and
Mathematical Programming:

Prof. dr. M. Uetz
Dr. ir. G.F. Post

Supervisors Tissue Regeneration:

Dr. J. de Boer
MSc. H.V. Unadkat
Dr. R.K. Truckenmüller



University of Twente
Enschede - The Netherlands

Preface

This report constitutes my Master of Science thesis in Applied Mathematics. The project I worked on is a collaboration between the Discrete Mathematics and Mathematical Programming (DMMP) group, and the Tissue Regeneration (TR) group, both part of the University of Twente, located in Enschede, the Netherlands. The project applies concepts from mathematics to a tissue engineering application. My Master of Science research was performed between June 1, 2008 and May 27, 2009.

At the start of my research, I did not know for sure which direction the research would take. Eventually it turned out that it included elements of fields as diverse as cell biology, materials science, statistical analysis, signal processing, and optimization algorithms. This made it a challenging project in which concepts from multiple fields were combined.

During my research Marc Uetz and Gerhard Post were my supervisors from the DMMP group. During our weekly meetings we had a lot of useful discussions on the mathematical parts of my research. These discussions helped me to better formulate my own ideas and to view the problems encountered during my research from another perspective. Jan de Boer, Hemant Unadkat, and Roman Truckenmüller were my supervisors from the TR group. During my stay at the TR group they helped me with the biological side of my research. When I started my research, my only biological education consisted of two years of weekly biology courses during high school. By pointing me to interesting papers and explaining biological concepts to me, they helped me build my knowledge on the biology relevant for my research. In addition, Hemant performed several biological experiments using the topographies I designed. I would like to thank all of my supervisors for their help and support.

Abstract

Cell behavior depends on multiple extracellular factors. One factor that influences cellular response is the surface topography of the surface on which the cells are positioned. This report describes strategies to find a topography that elicits desired cellular response best.

Using a parameterization of topographies, we design thousands of different topographies spanning a wide range of parameter values. Next we produce a PDLA + PLLA chip containing the designed topographies, called a TopoChip. We are unable to use the TopoChip to determine the cellular response for each of the topographies, because of technical problems.

We define an artificial quality measure and use this measure to determine the quality for each of the designed topographies. Using stepwise regression analysis we determine the characteristics of the topographies that influence their quality most. The stepwise regression algorithm is successful in identifying these characteristics, even when substantial noise is added to the quality measure.

We propose local search algorithms as a method to construct new topographies based on previously designed topographies and the measurements of their quality. Which implementation of a local search algorithm works well, is difficult to determine until cellular response becomes available.

Contents

1	Introduction	6
2	Biological and technical background information	8
2.1	Mesenchymal stem cells	8
2.1.1	Cell cycle	9
2.1.2	Mechanotransduction	10
2.2	The TopoChip	10
2.2.1	Material	11
2.2.2	Layout	11
2.2.3	Production process	11
2.3	Obtaining the cellular response	16
2.4	Realization	17
3	Formulation of the problem statement and plan of action	20
4	The first generation of features	21
4.1	Representation of features	21
4.2	Goal of the first generation of features	22
4.3	Design of the first generation of features	22
4.4	Choice of parameter values and ranges	23
5	Statistical model	26
5.1	Regression analysis	26
5.2	Candidate variables	27
5.2.1	Choice of the dependent variable	27
5.2.2	Choice of the independent variables	29
5.2.3	Discrete Fourier transform	31
5.3	Stepwise regression	36
6	Results	40
6.1	Results of the regression analysis	40
6.1.1	Dependent variable without noise	40
6.1.2	Dependent variable with noise	46
6.2	Summary of the results	48
7	Local search algorithms for improvement of features	49
7.1	Solution representation	50
7.2	Determining the quality of a solution	51
7.3	Implementation of the local search algorithm	52
7.4	Evolutionary algorithms	52

8	Conclusions	56
9	Recommendations	57
10	Bibliography	58
A	Parameter values and ranges	59
B	Independent variables used in the regression analysis	60

Chapter 1

Introduction

Stem cell therapies form a promising approach for improving the treatment of several diseases and medical conditions. For example, in [1] stem cells are collected from the bone marrow of a patient suffering from degenerative joint disease. Subsequently they are cultured outside the patient's body and afterwards injected into the patient's knee. After the treatment the patient's cartilage and meniscus show significant growth and the patient experiences less pain and increased range of motion. Influencing the behavior of stem cells can form the basis for new stem cell therapies.

Stem cell behavior like cell attachment, proliferation, and differentiation, is influenced by multiple extracellular factors. These factors include the composition and constitution of the extracellular matrix, the concentration of certain soluble factors in the cells' environment, and the topography of the surface on which the cells are positioned. In this research we focus on the surface topography. It has been shown in the literature that cellular response depends on the surface topography, see for example [2] and [3].

We consider $2\frac{1}{2}$ D topographies. These are two-dimensional topographies which have two different height levels. Since each possible two-dimensional black-and-white pattern can be transformed into a $2\frac{1}{2}$ D topography, the amount of possible topographies is enormous (even infinite when no restrictions on the resolution of the pattern are imposed).

The challenge is to find a topography that strongly induces the desired cellular response among the enormous amount of possible topographies. To search for such a topography we use high throughput screening (HTS). Using HTS thousands of topographies can be screened for their cellular response at the same time. Even using HTS there is a need for smart algorithms which identify which characteristics of the topography influence the cellular response and determine which topographies are included in the next generation of topographies that are screened for their cellular response.

This report starts with a chapter describing the biological and technical background of the research. We describe the cells used in the research, the TopoChip which is used for HTS of the topographies, and the measures that can be used as the cellular response. Afterwards the problem statement is formulated and we describe the plan of action. The next chapter deals with the design of the generation of features which is used in the first step of the HTS process. After that we formulate the statistical model which is used to describe the relation between the cellular response and the characteristics of the topographies. Following that we perform a stepwise regression analysis of the data for the first generation of topographies and describe the results. In the next chapter we propose algorithms that can be used to design a

new generation of topographies, using data from the previous generations. The last chapters contain the conclusions and recommendations.

Chapter 2

Biological and technical background information

This chapter contains a short overview of the biology and technology involved with our research. The goal of the chapter is to describe the materials and methods used to collect the data necessary for our research and to show the opportunities and limitations they present. The first section describes the mesenchymal stem cells used in our research. The second section is about the layout and production of the TopoChip. This is the polymer chip which contains the topographies. The next section deals with the collection of the cellular response of the cells which are cultured on the TopoChip. This cellular response is used as data in our research. In the final section we describe the realization of the data collection procedure described in the previous sections.

2.1 Mesenchymal stem cells

The cells that are used in our research are immortalized human mesenchymal stem cells (MSCs). In this section an introduction to these cells is given and some of their properties relevant to the research are presented.

MSCs are usually extracted from bone marrow. They typically have a small cell body and look long and thin. Their diameter is approximately between $10\ \mu\text{m}$ and $30\ \mu\text{m}$, though in some cases they can stretch to be even longer. For some pictures of MSCs, see Figure 2.1.

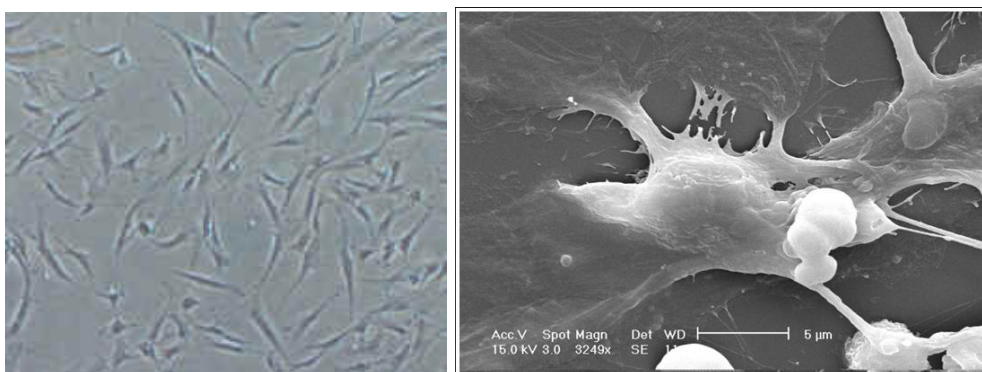


Figure 2.1: The picture on the left shows a colony of MSCs. The picture on the right provides a closer look at two MSCs.

Two main properties of stem cells are that they are able to reproduce through mitotic cell division (see Section 2.1.1) and can differentiate into other cell types. Differentiation is the process by which a less specialized cell changes into a cell with a more specialized function. MSCs are multipotent cells, which means that they can differentiate into a number of other cell types with different functionality, but this number is limited. Lineages (cell types) that MSCs can differentiate into include:

- Osteoblasts, which are responsible for bone formation.
- Chondrocytes, which can be found in cartilage.
- Myocytes, which can be found in muscles.
- Adipocytes, which store fat.

Before the cells are used in our research, they are immortalized. This is done to have the cells retain their multipotency and their ability to proliferate (reproduce). When cells are not immortalized, they may become unable to proliferate or differentiate into other cell types after a number of generations of cell divisions.

The reasons that MSCs are used in our research are that they typically proliferate fast, and because of their multipotency can form complex tissues that include multiple cell types.

2.1.1 Cell cycle

The cell cycle consists of the events leading to the replication of a cell. It contains two periods. In the interphase the cell grows to approximately twice its size, it accumulates nutrients and duplicates its DNA. During the mitosis (M) phase the cell splits into two separate cells. For a schematic overview of the cell cycle see Figure 2.2.

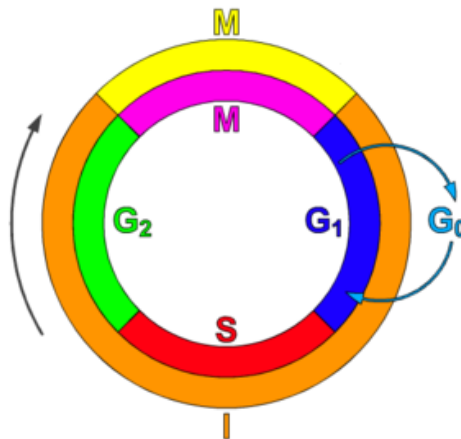


Figure 2.2: A graphical replication of the cell cycle. Depicted are the following phases: M (mitosis), I (interphase), G₀ (gap 0), G₁ (gap 1), S (synthesis), G₂ (gap 2).

In summary, the following happens during each of the phases:

- G₀ phase: this is the resting phase. The cell has (temporarily) stopped dividing, usually because the conditions for dividing are unsuitable, for example because of a lack of available nutrients. A cell usually enters the G₀ phase from the G₁ phase and it can reenter the cell cycle via the G₁ phase as well.

- G₁ phase: this is the part of the cell cycle in which the cell grows most. At the end of the G₁ phase there is a checkpoint to determine whether the DNA of the cell is intact and whether it is functioning normally.
- S phase: during this phase the DNA of the cell is duplicated.
- G₂ phase: during the G₂ phase the cell grows rapidly. At the end of the phase there is a checkpoint to determine whether the cell is ready to proceed to the M phase and divide. This prevents damaged DNA from being replicated.
- M phase: during the M phase a cell divides. All cell parts are distributed approximately equally over the two daughter cells. The mother and the two daughter cells are genetically identical to each other.

2.1.2 Mechanotransduction

In our research we try to influence cellular response by exposing cells to different surface topographies. These topographies cause the cells to be exposed to mechanical forces. In [4], mechanotransduction is defined as the process of converting mechanical forces into biochemical forces and integrating these signals into the cellular responses. It is well-known that mechanical forces on cells can influence the cellular response. The mechanisms describing the conversion of mechanical forces into biochemical signals however, are still undefined.

The results of influencing cellular response by applying mechanical forces, include:

- In [5] MSCs are forced to take on certain shapes by putting single cells on small islands of materials where they can attach to. The regions surrounding these islands are made non-adhesive. On large islands, where cells are able to spread and flatten, they mostly differentiate into osteoblasts. On small islands, where cell spreading is prevented, the cells mostly differentiate into adipocytes.
- In [2] immature osteoblast precursor cells derived from fetal rat calvariae are cultured on strips with different roughness. The proliferation and expression of alkaline phosphatase (ALP) depends on the roughness of the strips. ALP expression is an indication for differentiation to osteoblasts. Proliferation and ALP expression are highest for an average roughness (average distance from points on the roughness profile to an arithmetic center line) of 0.8 μm .
- In [3] nanotopographies are created by making 120 nm deep pits in a polymethylmethacrylate (PMMA) substratum. The distance between pits is on average 300 nm . Different topographies are used, including arranging the pits in hexagonal arrays, in square arrays, in square arrays with random displacement of the pits by up to 50 nm , and by creating the pits at random positions. Of these topographies, MSCs planted on the square arrays with displacement show most signs of differentiating into osteoblasts.

2.2 The TopoChip

In our research MSCs are cultured on a polymer chip containing different topographies, called a TopoChip. After culturing the cells, the cellular response for each of the topographies is measured. In this section the material used for the TopoChip, the layout of the TopoChip, and the production process of the TopoChip are described.

2.2.1 Material

The TopoChip is made from a mixture of two forms of polylactide (PLA) called poly-L-lactide (PLLA) and poly-D-lactide (PDLA). PLA is produced using renewable resources like sugarcane and cornstarch. The reasons that PDLA + PLLA is used, include:

- It is bioresorbable, which means that in the human body it degrades and is completely eliminated via natural pathways.
- It is commonly used in tissue engineering applications.
- It has sufficient mechanical strength for most tissue engineering applications, including engineering of skin, muscle, and blood vessel tissue.

2.2.2 Layout

The TopoChip is subdivided in TopoUnits. These TopoUnits are $290\ \mu\text{m}$ by $290\ \mu\text{m}$ square areas in which the topographies are created. Between TopoUnits and at the edges of the TopoChip there are walls that are $10\ \mu\text{m}$ thick and $50\ \mu\text{m}$ high. The TopoChip is square, with 66 TopoUnits on each side. This makes the total number of TopoUnits on the TopoChip equal to 4356. One side of the TopoChip is almost 20 mm long.

Each TopoUnit contains a replicated feature. A feature is a square area containing a certain topography with a height of $10\ \mu\text{m}$. The feature is copied enough times that it fills the TopoUnit. There is no distance between features (empty space may occur, however, if the feature contains empty space at its edges), but there is an empty space of $5\ \mu\text{m}$ between the walls and the features. This is done to prevent the walls and the features from interfering with each other when there are small inaccuracies during the process of imprinting the features and walls on the TopoChip. The reason that only one feature is used in one TopoUnit, is to try to have all cells placed in the same TopoUnit experience the same topography. For an example of the layout of part of a TopoChip, see Figure 2.3. This figure shows several TopoUnits, each containing its own feature.

The lower left quadrant (33 by 33 TopoUnits) of the TopoChip is the same as the upper right quadrant. Similarly, the lower right quadrant is the same as the upper left quadrant. This way, each feature appears in two different TopoUnits. This is done for two reasons. First of all, it may prove that TopoUnits containing the same feature often provide very different cellular response. In this case it is probably not possible to find reliable correlation between feature characteristics and cellular response. Also, the position of a TopoUnit on the TopoChip may influence cellular response. If this seems to be the case, modifications to the setup of the experiment have to be made in the future.

Four of the TopoUnits do not contain any features at all (alternatively, one could say they contain flat features). This is done to be able to compare the cellular response of TopoUnits containing features with the cellular response of TopoUnits that do not contain any features.

2.2.3 Production process

The TopoChip is created in three steps. These steps are detailed in this section. First, two masks are made that contain the design for the topographies. There is one mask

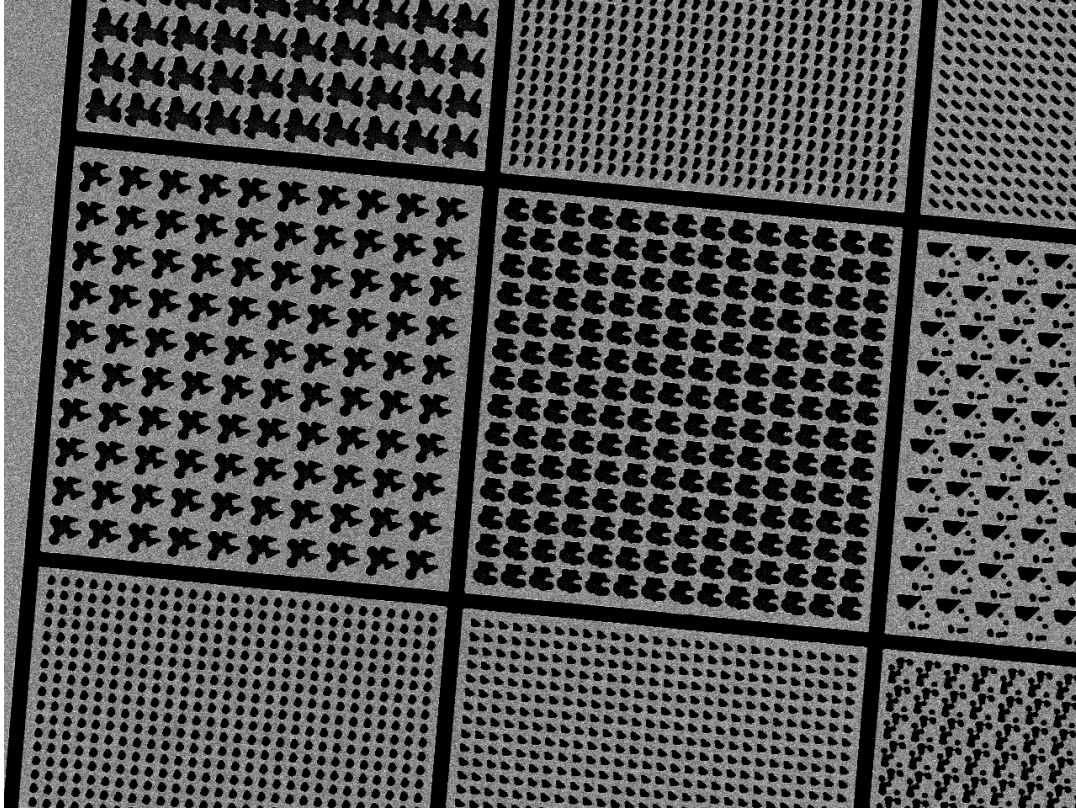


Figure 2.3: Layout of part of a TopoChip

for the walls of the TopoUnits and one for the features. Afterwards, a silicon wafer containing the topographies is made using the masks. The last step is making a PLA TopoChip using the silicon wafer.

Step 1: Creating the masks

The first step of the creation of the TopoChip is creating the masks. Since the walls and the features have different height, separate masks are needed for the walls and the features. The masks are later used to create silicon wafers. This process can be repeated multiple times. The process of creating the masks is the most time-consuming of all the steps.

At the start of the process, the mask consists of a layer of soda-lime glass (standard glass), a thin layer (about 90 nm thick) of chromium on top of that, and a thin layer of resist on top. A laser is used to transfer the design of the topographies. Part of the resist is exposed to the laser (shown in red in Figure 2.4). The mask is then put in developing fluid which removes only the resist exposed to the laser. The chromium is now exposed at these places. Afterwards, the exposed chromium is etched using wet etching. Finally, the remaining resist is removed. See Figure 2.4 for a pictorial representation of this process.

Step 2: Creating the silicon wafer

When the masks are finished, the designs on the masks are transferred to a silicon wafer. Separate masks are used for the design of the features and the design of the walls.

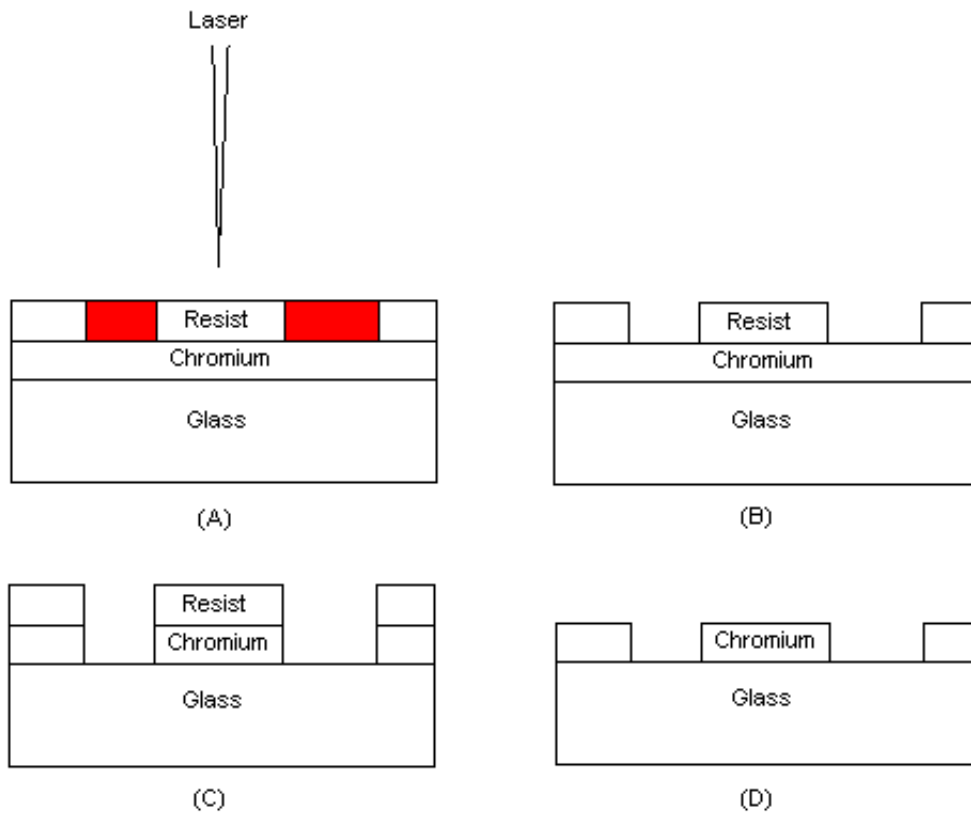


Figure 2.4: Creation of the masks. In (A) the topographies are transferred to the resist with a laser. In (B) the part of the resist exposed to the laser is removed using resist developer. In (C) the regions of the chromium that are exposed, are etched using wet etching. In (D) the remaining resist is removed.

At the start of the process of transferring the designs on the mask to the wafer, the wafer consists of a layer of silicon that is $525\text{ }\mu\text{m}$ thick. In a thermal oxidation step, the top 500 nm of the silicon is oxidized to form a thin layer of SiO_2 . The SiO_2 layer is added, because the resist layer does not sufficiently protect the covered silicon during the etching step, while the SiO_2 layer does. On top of the SiO_2 layer a thin layer of resist is deposited.

The mask for the features is placed above the wafer with the chromium parts at the bottom, at a position as close to the wafer as possible. The mask is exposed to ultraviolet (UV) light from above. The glass layer of the mask is very transparent to this light, while the chromium layer allows virtually no light at all to pass. This causes the parts of the resist not protected by the chromium to be exposed to the UV light. The exposed part of the resist is subsequently removed with a developing fluid. Afterwards, the exposed parts of the SiO_2 layer are etched by reactive-ion etching (RIE). This is an etching technology that uses chemically reactive plasma to remove material. The remaining resist is removed and a new layer of resist is put on top of the wafer. The mask for the walls is used to expose the resist that has to be removed to UV light. Next the exposed resist is removed and the exposed SiO_2 is etched. At this point only the part of the silicon where the walls have to be is exposed. Using RIE the silicon is etched to form the walls. The etching is done to a depth equal to the needed height of the walls minus the height of the features. During a later etching step in which the features are etched, the walls are etched as well so they obtain the proper height. The resist is removed again, so that the silicon is exposed at the places where the features have to be. Afterwards, the silicon is etched. This etches the features in the wafer and etches the walls to the proper depth. Finally, the remaining SiO_2 is removed. See Figure 2.5 for a pictorial representation of the process.

Because it is difficult to etch deep straight structures in silicon without etching any silicon at the sides of the desired structure, the etching of the silicon actually happens in steps. In each step a small amount of silicon is etched and the sides of the structure are covered by a passivation layer, which prevents etching of the sides. When the desired structure is etched, the passivation layer is removed again.

Step 3: Imprinting the TopoChip

The final step is to imprint the topographies in the TopoChip using the silicon wafer. This is done by hot embossing. A sheet of PDLA + PLLA is placed on top of the silicon wafer. This polymer sheet is heated to just above its glass transition temperature. This causes the polymer to become soft. By applying pressure the polymer sheet is pressed into the wafer, causing it to fill the indents in the wafer. Afterwards the wafer and polymer sheet cool down and the polymer sheet is demolded from the wafer. The polymer sheet now contains the topographies and constitutes the TopoChip.

Loss of accuracy

During the production process of the TopoChip there are several moments at which accuracy is lost while transferring the topographies. First of all, the laser used to produce the masks has a spot size and step size. The pattern describing the topographies is built up by printing partly overlapping spots with the laser and this does not transfer the pattern 100% accurately. In addition, the developing fluid does not remove exactly the resist exposed to the laser. At some places too much resist may

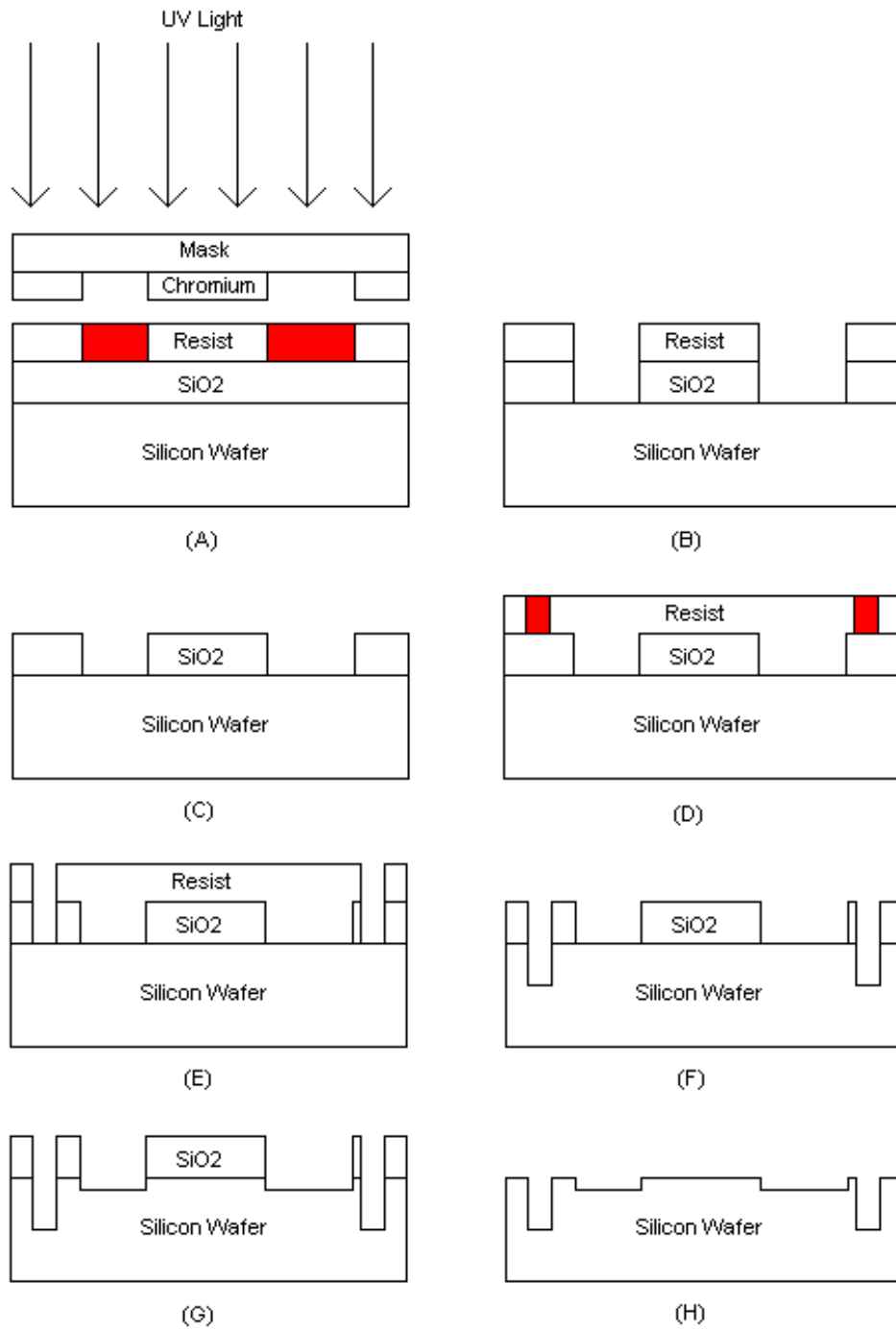


Figure 2.5: Creation of the wafer. In (A) UV light is used to transfer the design of the features from the mask to the resist. In (B) the resist exposed to UV light is removed and the exposed SiO_2 is etched. In (C) the remaining resist is removed. In (D) new resist is put on and the pattern for the walls is transferred to the resist. In (E) the resist exposed to UV light is removed and the exposed SiO_2 is etched. In (F) the exposed silicon is etched and the remaining resist is removed. In (G) the exposed silicon is etched. In (H) the remaining SiO_2 is removed.

be removed and at some places too little. Also, the etching of the chromium may remove too much or too little chromium at some places.

In the second step some of the deep UV light that passes through the gaps in the chromium layer may be partly diffracted in other directions by physical effects. The intensity of the diffracted light is higher for certain angles, depending on the wave length of the light and the size of the gap (see Figure 2.6). This happens primarily at the borders of gaps in the chromium layer. This effect may cause part of the resist to get exposed to the light even though it is covered by chromium. In this step it can also happen that too much/too little resist is removed or too much/too little silicon is etched, similar to the first step.

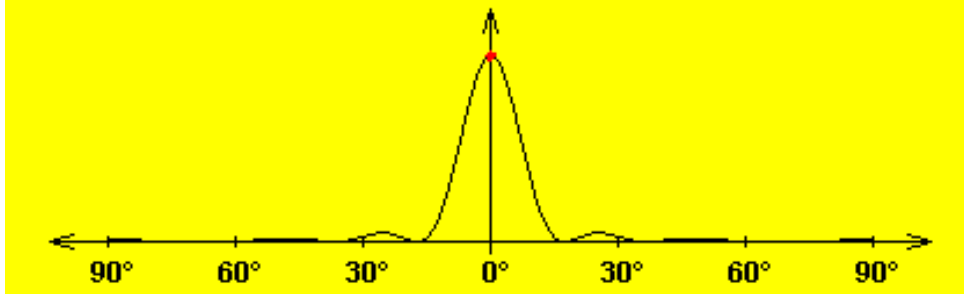


Figure 2.6: Example intensity profile of diffracted light.

During the third step the temperature varies. Since the wafer and polymer shrink and expand with temperature changes at a different rate, when the TopoChip is removed from the wafer it may have a slightly different shape than the wafer. In addition, because the polymer does not fill all of the corners of the indents in the wafer completely, some of the angles in the TopoChip may be more rounded than on the wafer.

The inaccuracy in the complete process is expected to produce inaccuracies of several hundreds of nanometers. Using different (and more expensive) materials and techniques it is possible to reduce the size of the inaccuracies.

Since there are significant inaccuracies in the production process, it is undesirable to have the topography contain elements of small size. In addition, when the topography contains small elements it is harder to demold the polymer sheet without breaking it or the silicon wafer. For these reasons, we make the choice to restrict the size of elements of the topographies. Each element (part of a topography surrounded by empty space) has a minimum diameter of $3\text{ }\mu\text{m}$.

2.3 Obtaining the cellular response

Cells are seeded on the TopoChip such that each TopoUnit contains a number of cells. Preferably, this is done such that the number of cells in each of the TopoUnits is about equal, because cell density can influence the cellular response. After a period of culture on the different topographies, the cellular response of the cells in that unit is measured for each TopoUnit. There are multiple possible choices for the cellular response to measure. Some of the possibilities include:

- *The fraction of cells that reenters the cell cycle.* Before the cells are seeded on the TopoChip, a number of MSCs is starved for fetal bovine serum (FBS). FBS is often used for culturing MSCs. The proteins contained in FBS stimulate cell survival, growth and proliferation. By culturing the cells in medium that does

not contain FBS for a number of days, cells leave the cell cycle and enter phase G_0 (see Section 2.1.1 for more information on the cell cycle). During a period of culture on the TopoChip, some of the cells reenter the cell cycle. Measuring the total number of cells and the number of cells that reenter the cell cycle allows the computation of the fraction of cells that reenters the cell cycle.

- *Cell morphology.* The morphology of cells can be characterized by multiple measures like the aspect ratio (ratio between cell width and length) or the perimeter of the cell. Another commonly used measure is the form factor FF , defined as $FF = \frac{4\pi \cdot A}{P^2}$. A is the cell area and P is the cell perimeter. For a perfectly circular object $FF = 1$ and for long, thin objects, FF is close to 0. In addition, there are numerous other ways to define the cellular response using characteristics of the cell morphology.
- *Differentiation.* MSCs can differentiate into different lineages. The selected cellular response can be based on the ratio of the different lineages that the MSCs differentiate into.

The most suitable cellular response to measure depends on the goal of the topographies. The measure chosen should be related to the response that is desired from the cells. In our research the choice of the cellular response is still open. A definitive choice is not yet made, because of several problems that are encountered during the production of the TopoChip and the seeding of the cells on the TopoChip. These problems prevent culturing cells on the different topographies and measuring the cellular response. For more information on the problems, see Section 2.4.

2.4 Realization

During the creation of the TopoChip and the seeding of the cells several problems are encountered. The first problem is that the features and walls in the TopoChip do not have the required shape and height. A picture of a part of the TopoChip can be seen in Figure 2.7. By design, the edge of the features should have the same height as the center. Also, the height of the walls and the features should be much larger than it is. The reason that the features do not have the required size and shape is that the cavities in the silicon wafer are not completely filled during the hot embossing step of the production of the TopoChip. See Figure 2.8 for a pictorial representation. It is possible to increase the degree in which the cavities are filled by increasing the pressure or temperature used during the hot embossing step. When the cavities of the wafer are better filled however, it gets harder to demold the polymer sheet. Trying to demold the polymer sheet, parts of the wafer and the sheet break. To solve this problem, it is necessary to use an alternative production process of the TopoChip. Another possible approach is to decrease the height of the walls and the topographies and/or increase their width and length. This makes them stronger and makes it easier to demold the polymer sheet from the wafer.

A second problem arises because the walls are too low. During attempts to seed the cells on the TopoChip, cells get placed on positions where they are on top of a wall and partly in different TopoUnits. This way they cannot be categorized as belonging to a certain TopoUnit and the influence of the topography on their cellular response cannot be accurately measured.

Since the time available for our research is limited, and the problems concerning the creation of the TopoChip and a proper seeding of the cells cannot be expected to

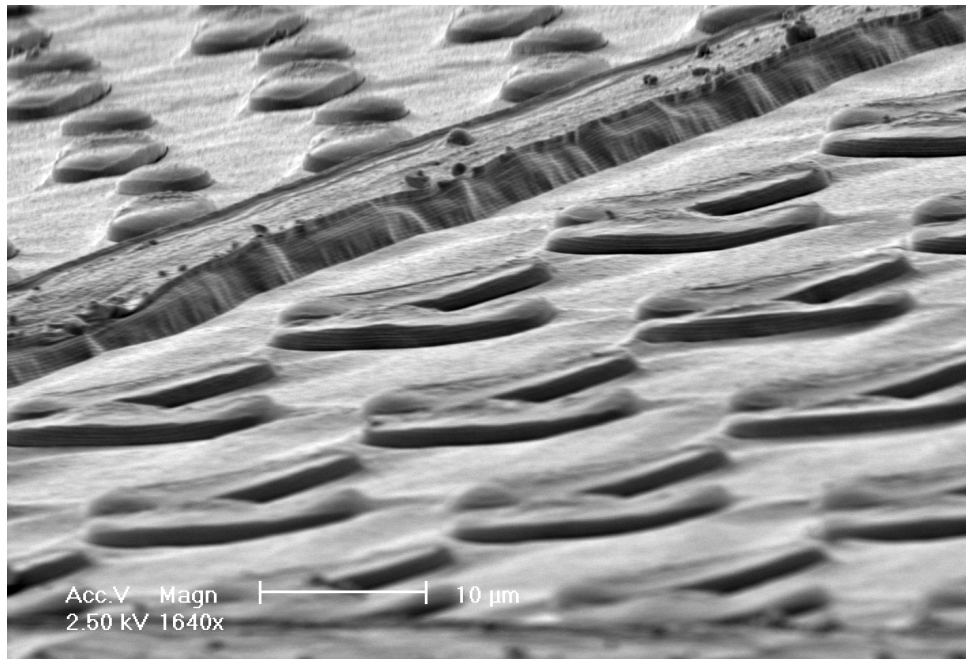


Figure 2.7: A picture of a part of the TopoChip. Note that the walls and the features do not have the required height of $50\ \mu\text{m}$ and $10\ \mu\text{m}$ respectively, and that the shape of the features is rounded.

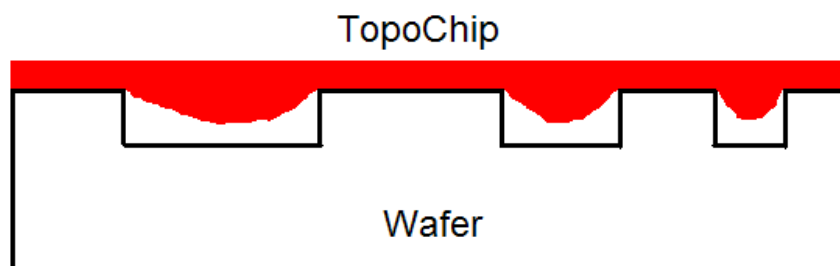


Figure 2.8: During the thermoforming process the cavities in the silicon wafer are not completely filled by the polymer. This causes the features on the TopoChip to be too small and rounded.

be solved in a short amount of time, we make the choice to abandon the attempts to collect data on the cellular response. Instead, we create artificial data for the cellular response for each of the topographies in order to try our algorithms. If at a later stage data for cellular response for each of the topographies becomes available, the algorithms can easily be applied to this data. The choice for the artificial data to use is described in Section 5.2.1.

Chapter 3

Formulation of the problem statement and plan of action

The goal of our research is to design a strategy to find a feature with as large a value for the cellular response as possible. This strategy is composed of the following steps:

- We create a first generation of features. Each feature is characterized by a number of parameters. We try to design the features such that there is a lot of variation in the values for these parameters between features. These features are used to produce a TopoChip and for each of the features the cellular response is determined.
- We define several variables that characterize a feature and compute their value for each of the features. Using stepwise regression analysis we determine which of the variables influence the cellular response most (note that we use artificial data for the cellular response, see Section 2.4). The most influential variables can be used to decide on a suitable representation of a feature.
- We propose ideas for local search algorithms that can be used to design features for the next TopoChip, based on the measured cellular response for previously tested features. We discuss some of the problems and opportunities present when using local search algorithms in our research.

Chapter 4

The first generation of features

This chapter starts with an introduction to the representation of features. Afterwards, the goal of the first generation of features is described. Next, the way in which the features are designed is described. Finally, some comments are given on the ranges and values for the parameters that are used during the construction of the features.

4.1 Representation of features

In order to do any optimization at all, it is necessary to have structure in the space of all possible features. Without any structure, searching for the best feature would just come down to randomly trying possible features. With a structure, knowledge of the results from previously tested features can be used to select new features to try, in a smart way.

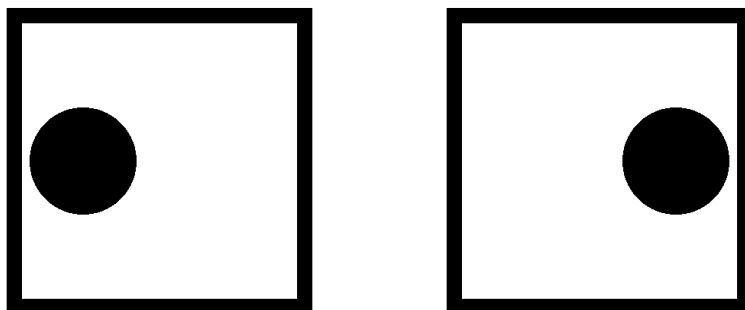


Figure 4.1: Two example features

There are multiple possible representations of features. Take the left feature in Figure 4.1, for example. It can be represented by discretizing it and specifying for each pixel whether it is black or white. Alternatively, it can be represented by describing the black part as a circle with a certain radius and center position. Another possibility would be to represent it by the Fourier coefficients obtained after applying a two-dimensional Fourier transform to the feature. In addition to these representations, there are numerous alternatives.

The choice of representation influences which features are similar, i.e. close to each other in the space of all possible features with regard to some distance function. For example, when a pixel representation is chosen, the two features in Figure 4.1 are not very similar. A lot of white pixels in the left feature are black in the right feature,

and vice versa. However, both features can be described as a circle with the same diameter and vertical position, so in this sense they can be regarded as very similar. If you take symmetry into account, you can even consider the two features as exact copies.

It is hard to say in advance which representation works best. From an optimization point of view, the ideal representation is one for which features that are similar, provide similar cell response. Since we do not want to make many assumptions on which characteristics of features influence the cell response, research is necessary to decide on a suitable representation.

At some point it is necessary to reconstruct a feature from its representation. Ideally, the representation of a feature should be extensive enough that the reconstruction provides an (almost) identical copy of the feature. On the other hand, for some algorithms it is preferable to have compact representations of features, because this may result in discovering good features faster. A balance between these two conflicting requirements has to be found. **It may be best to let go of a one-to-one correspondence between features and their representations, as long as features that have the same representation are similar.**

4.2 Goal of the first generation of features

The goal of the first generation of features is to find a suitable representation for features. This means that we want to describe features with a collection of parameters that have a high correlation with the cell response. This representation can subsequently be used in algorithms that try to discover better features.

Note that having the best possible features in the first generation of features is not a goal. It is more useful to have a spread over bad and good features. When the difference in cell response of the features is bigger, it becomes easier to correlate parameters and cell response.

4.3 Design of the first generation of features

Preferably, the first generation of features should contain features that are very different. As mentioned in Section 4.1, whether features are different or similar depends on the representation that is used. Because no assumptions are made on which representation is most useful, it is not possible to guarantee that the features in the first generation are very different from each other with regard to the most useful representation.

The choice is made to base the design of the first generation of features on the idea that a lot of different shapes can be approximated reasonably well with a collection of circles, triangles, and lines (thin rectangles). The circles can be used for large areas, the triangles for (sharp) corners, and the lines for long, thin areas. For an example, see Figure 4.2. By taking the opposite direction of generating features as a collection of circles, triangles, and lines we try to generate a collection of features that have shapes that differ a lot from each other.

In the following, when a parameter is assigned a value, this is done by drawing a **uniformly distributed** number from the range of values that the parameter is allowed to take, unless otherwise specified. Some remarks about the ranges for all of the parameters can be found in Section 4.4. The exact ranges can be found in Appendix A.

The construction of a feature is as follows:

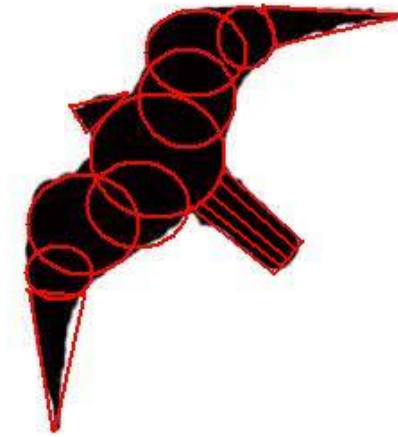




Figure 4.2: The shape of a falcon, approximated by a collection of circles, triangles and rectangles

- The size of the feature is determined.
- The number of primitives composing the feature is determined.
- Each primitive type (circles, triangles, lines) is included in the feature with probability 0.5. If this results in a situation where no primitive types are included, this step is repeated.
- A division of the primitives over the primitive types is determined. This is done in such a way that each **division**  it consists of at least one primitive of the primitive types selected in the previous step and does not include any primitives of types not selected in the previous step, is as likely to be chosen.
- The diameter of circle primitives, the shortest side of triangle primitives, and the length of line primitives are determined. Note that these values are the same for all of the primitives of that type in the feature.
- A standard deviation for the rotation of primitives is determined.
- One at a time, the primitives are placed in the feature. First the rotation of a primitive is determined. This is done by drawing a value from a **normal distribution**  with mean 0 and standard deviation equal to the standard deviation determined in the previous step. A rotation of 0 means that a triangle primitive has its sharpest corner pointing to the right and a line primitive is in a horizontal position. The positive direction of rotation is counterclockwise. For circle primitives, the rotation is irrelevant.

When the rotation is known, all possible positions where the primitive can be placed such that it is entirely within the feature are determined. One of these positions is uniformly selected and the feature is placed. Note that it is irrelevant whether the feature overlaps with other features or not.

4.4 Choice of parameter values and ranges

Some of the parameters used to construct the features have the same value for all of the features. Other parameters take on different values for each feature. These

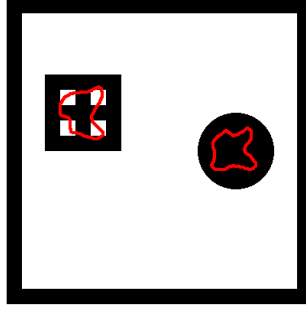


Figure 4.3: On large features, cells may experience a very different surface, depending on where they are placed on the feature

parameters have a lower and upper bound, between which they take values. The lower and upper bounds may depend on the feature size. The choice of parameter values and ranges is based on the following ideas and limitations:

- Because of the technical limitations mentioned in Section 2.2.3, the ‘diameter’ of a primitive has to be at least $3\ \mu m$. This is enforced by setting the minimum diameter of a circle primitive, the minimum length of the shortest side of a triangle primitive, and the thickness and minimum length of a line primitive equal to $3\ \mu m$. The maximum thickness of a line primitive is set to $3\ \mu m$ as well. This is done to make the lines as thin as possible. Still, some lines look more like rectangles, because of the minimum thickness requirement.
- There are three different feature sizes: $10\ \mu m$ by $10\ \mu m$, $20\ \mu m$ by $20\ \mu m$, and $28\ \mu m$ by $28\ \mu m$. All of these sizes are chosen in the same order of magnitude as the size of a cell. Much smaller feature sizes are not possible, because of the technological limitations mentioned above. Much larger feature sizes are undesirable, because in this case cells that are on the same feature, may experience a very different surface. This effect makes it more difficult to determine the quality of a large feature. See Figure 4.3 for an example.

The exact feature sizes are chosen because they fit an integer number of times in the area of the TopoUnit on which placement of features is allowed, which is $280\ \mu m$ by $280\ \mu m$ (see Section 2.2).

- The number and size of primitives are chosen in a way that there are both features that are pretty sparse and features that are pretty dense. This means that the upper bounds for the number and size of primitives are higher for larger features than they are for smaller features.
- The top angle of a triangle primitive (opposite of the shortest side) is always equal to 36° . The other two angles are both equal to 72° . These values are chosen to have some sharp and some less sharp angles in the feature. Because primitives may overlap angles of other sizes may occur in a feature.

A table with all values and ranges for parameters used to construct features can be found in Appendix A. In Figure 4.4 four example features are shown.

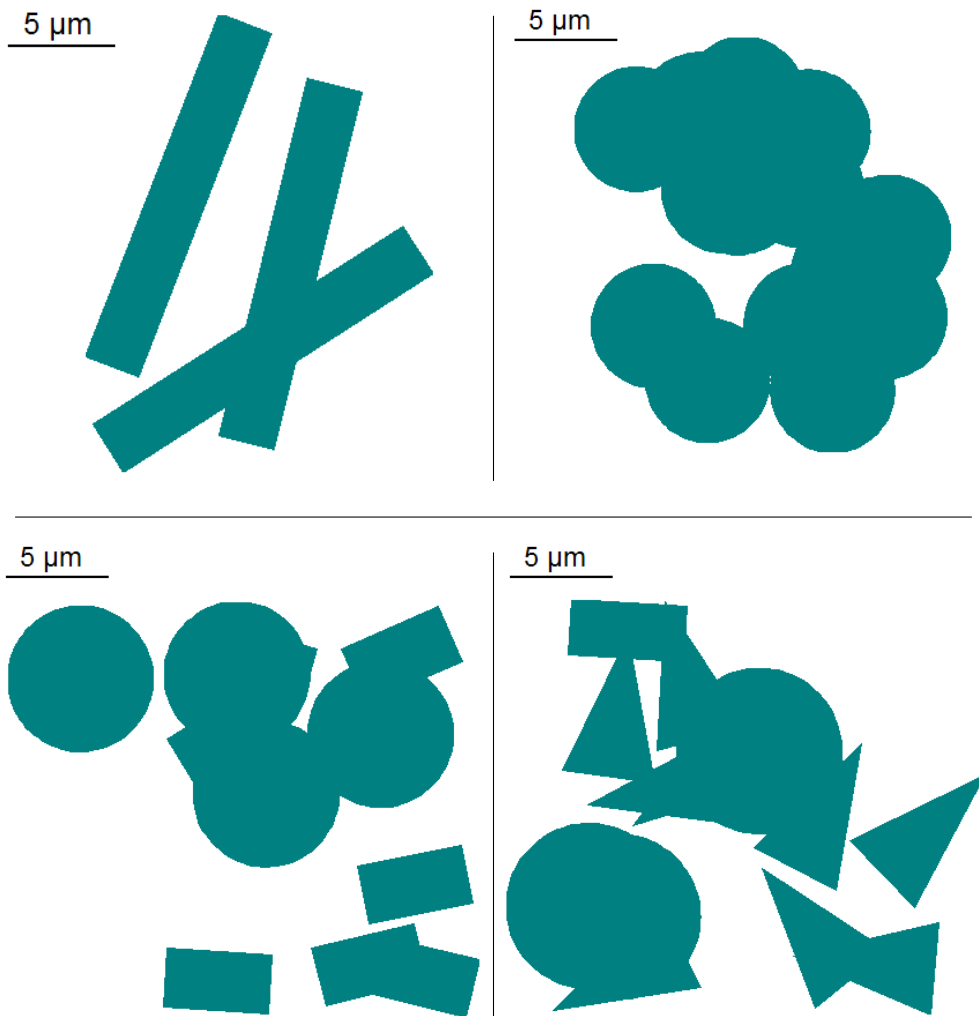


Figure 4.4: Four example features.

Chapter 5

Statistical model

This chapter starts with an introduction to regression analysis. Afterwards the variables which are considered for inclusion in the regression model are described. Finally, the stepwise regression algorithm used to determine characteristics of features that influence cellular response, is described.

5.1 Regression analysis

Regression analysis is a collection of techniques to estimate the relationship between a number of variables. In the case of linear regression, the observed value Y_j of the independent variable Y is modeled as a function of the observed values x_{ij} of a number of independent variables x_i , plus a noise term ϵ_j :

$$Y_j = \beta_0 + \beta_1 x_{1j} + \beta_2 x_{2j} + \dots + \beta_k x_{kj} + \epsilon_j \quad (5.1)$$

In Equation (5.1), the β_i are unknown coefficients and ϵ_j is a random variable that represents a noise term. Equation (5.1) is called linear, because it is linear in the unknown coefficients β_i . It does not have to be linear in the dependent variables x_i . The goal of regression analysis is to find estimates $\hat{\beta}_i$ for the coefficients β_i that best fit the collection of available data for Y and the variables x_i .

Suppose that the following assumptions hold for the noise terms ϵ_j :

- $E(\epsilon_j) = 0, \forall j$. All errors have mean 0.
- $Var(\epsilon_j) = \sigma^2, \forall j$. All errors have the same variance.
- $Cov(\epsilon_i, \epsilon_j) = 0, i \neq j$. Errors are uncorrelated.

In [6] it is shown that if the above assumptions hold, the ordinary least squares (OLS) estimator is a best linear unbiased estimator (BLUE) for the unknown coefficients β_i . OLS tries to minimize the sum of the squares of the residuals r_j , see Equation (5.2).

$$S = \sum_{j=1}^n r_j^2 \quad (5.2)$$

The residual r_j for case j of the data is equal to the measured value for the dependent variable Y_j minus the predicted value \hat{Y}_j , see Equation (5.3).

$$r_j = Y_j - \hat{Y}_j \quad (5.3)$$

The value \hat{Y}_j is computed using the independent variable values $x_{1j}, x_{2j}, \dots, x_{kj}$ and the estimated coefficients $\hat{\beta}_i$, see Equation (5.4).

$$\hat{Y}_j = \hat{\beta}_0 + \hat{\beta}_1 x_{1j} + \hat{\beta}_2 x_{2j} + \dots + \hat{\beta}_k x_{kj} \quad (5.4)$$

5.2 Candidate variables

5.2.1 Choice of the dependent variable

As mentioned in Section 2.4 we have not been able to collect the cellular response for the different topographies. Still it is interesting to see whether the regression analysis succeeds in identifying variables that are relevant for the quality of a feature. Since we cannot use the cellular response as a measure for the quality of a feature, we choose to create an **artificial quality measure** and treat this measure as the cellular response in the analysis. The chosen measure is the fraction of the feature covered by primitives (FCP). We choose this measure, because we expect it to depend on the characteristics of the feature used as the independent variables (defined in Section 5.2.2), while it is not clear in advance how exactly it depends on these characteristics, just like the cellular response.

FCP is computed by first discretizing the feature. The feature is represented as 100 by 100 pixels. Each pixel is considered covered by a primitive if the center of the pixel is inside one of the primitives. *FCP* is defined as the number of pixels covered by primitives divided by the total number of pixels.

There are reasons not to use *FCP* directly as the dependent variable, but to first apply a transformation to it. This transformation makes it more suitable for linear regression. The variable *FCPLOG* that is used as the dependent variable in the regression analysis is defined in terms of *FCP* as in Equation (5.5):

$$\text{LOGFCP} = \ln\left(\frac{\text{FCP}}{1 - \text{FCP}}\right) \quad (5.5)$$

To make sure that *LOGFCP* is well defined for $\text{FCP} = 0$ and $\text{FCP} = 1$, we round all values for *FCP* smaller than 0.01 to 0.01 and all values larger than 0.99 to 0.99. Rewriting Equation (5.5) to express *FCP* in terms of *FCPLOG*, provides Equation (5.6):

$$\text{FCP} = \frac{1}{1 + e^{-\text{FCPLOG}}} \quad (5.6)$$

Plotting *FCPLOG* against *FCP* shows a logistic curve, see Figure 5.1. A logistic curve is characterized by a period of slow growth, then a period of fast growth, after which the growth slows down again. The reasons that the transformation is applied to *FCP*, include:

- In cell biology, the relationship between cellular response *R* and a variable *X* that influences it (while keeping other variables constant) often takes the shape of a sigmoid. For small values for *X*, *R* is small as well. Then a critical value for *X* is reached and *R* increases fast. Finally, a maximum level for *R* is reached and it stays at about the same level, even if *X* is increased. This behavior of *R* makes a linear model of *R* in terms of *X* unsuitable. Since a typical plot of *X* against *R* resembles the plot in Figure 5.1 quite well, the proposed transformation in Equation (5.5) seems reasonable.

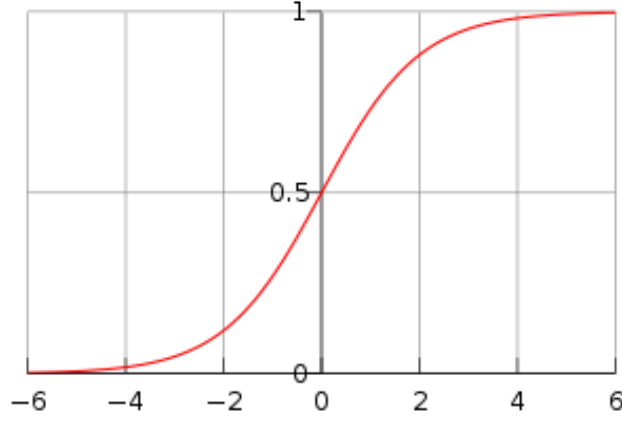



Figure 5.1: A plot of $FCP = \frac{1}{1+e^{-FCPLOG}}$. Note the period of slow growth, followed by a period of fast growth and another period of slow growth. This behavior is characteristic for a logistic curve.

- FCP can be interpreted as the probability that a pixel is covered by a primitive (or alternatively, the probability that a cell shows a positive response if real cellular response is used). FCP only takes values between 0 and 1. This is another reason that FCP is unsuitable to use as the dependent variable in linear regression, because for certain values of the independent variables x_i , the model can predict a value for FCP that is smaller than 0 or greater than 1. This problem does not occur when $FCPLOG$ is used as the dependent variable, because any value for $FCPLOG$ gives FCP a value between 0 and 1, see Equation (5.6). 

To get a feel for how the β_i can be interpreted after applying the transformation to FCP , Equation (5.5) can be rewritten as:

$$e^{FCPLOG} = \frac{FCP}{1 - FCP} \quad (5.7)$$

$$e^{\beta_i x_i} e^{FCPLOG} = \frac{FCP}{1 - FCP} \quad (5.8)$$

Equation (5.7) is derived by taking the exponential of Equation (5.5). In Equation (5.8) one of the β_i is separated from $FCPLOG$. $FCPLOG$ is equal to $FCPLOG - \beta_i x_i$. The right side of Equation (5.8) represents the odds of a pixel being covered by primitives, that is the ratio between the fraction of pixels covered by primitives and the fraction of pixels not covered by primitives. It can now be seen that one unit increase in x_i (while keeping other variables equal) changes the odds of a cell being covered by primitives by a factor e^{β_i} .

In a biological experiment there can be substantial noise. In our experiment for example, two TopoChips containing the same features can differ slightly because of small errors during the production process. Also, the cells are all unique and can behave differently. In addition, when the cells are placed on the TopoChip they are not always placed on exactly the same position on the feature, which can lead to different cell behavior. Because of this noise, measuring cellular response can provide different values for the same feature when the experiment is repeated. It is interesting to see whether the noise prevents the stepwise regression algorithm from identifying the significant independent variables. To get an impression of the influence of the

noise, we create new variables $FCPN_x$ by adding a noise term to FCP . Again we make sure that $FCPN_x$ takes values between 0.01 and 0.99. The variables $FCPN_x$ are defined as in Equation (5.9).

$$FCPN_x = \max(\min(FCP + \epsilon, 0.99), 0.01) \quad (5.9)$$

In Equation (5.9) ϵ is a normally distributed noise term with mean 0 and standard deviation x . As before we transform $FCPN_x$ to get the dependent variable $LOGFCPN_x$, see Equation (5.10).

$$LOGFCPN_x = \ln\left(\frac{FCPN_x}{1 - FCPN_x}\right) \quad (5.10)$$

We perform the regression analysis for the situation without noise, and for noise levels $x = 0.1$ and $x = 0.3$.

5.2.2 Choice of the independent variables

In this section a collection of variables that can be used as independent variables in the regression analysis is created. Whether these variables actually appear in the regression equation, depends on their significance in the stepwise regression process, described in Section 5.3. Some of the variables are based on the parameters that are used to create the features (see Appendix A for an overview of these parameters). Others are based on characteristics of the feature that cannot be computed directly from the parameter values. The reason to include a certain variable, is to investigate whether it has an influence on the cellular response.

In the rest of this section, the variables that are considered as independent variables in the regression analysis are listed. Each variable is given a name and the reason that it is included is given. For a short summary of which variables are included, see Appendix B. For a more extensive overview see below:

- The density of triangle primitives in a feature DT , the density of circle primitives in a feature DC , and the density of line primitives in a feature DL are included as possible independent variables. DT is computed by dividing the total number of triangle primitives in the feature by the feature area in μm^2 . DC and DL are computed analogously. These variables are included to research whether the type of primitives used to create the feature influences the cellular response. Each of the primitive types has its own characteristic. The line primitives are long and thin, the triangle primitives are pointy, and the circle primitives are round. Features composed of a lot of primitives of a certain type, usually share this characteristic with the primitive. We make the choice not to use the absolute amount of primitives of a certain type as an independent variable, but to scale it with the feature size. This is done because intuitively it seems that the density of primitives of a certain type is a more natural measure to compare features of different sizes than the absolute number of primitives of that type.
- It is possible that the size of the primitives present in a feature influences the cellular response. The size of a certain primitive type is unsuitable as an independent variable however, because it is meaningless for features that do not contain any primitives of that type. This effect obscures the influence of the variable on the cellular response. For this reason the choice is made to include as independent variables the number of primitives per μm^2 of a certain type in a feature times the area in μm^2 of that primitive type. This way the size of a

primitive type that is not contained in a feature has no influence on the variable value, since the product is zero. The following variables are included:

- TA , the product of the density of triangle primitives (DT) in a feature and the area of a triangle primitive.
 - CA , the product of the density of circle primitives (DC) in a feature and the area of a circle primitive.
 - LA , the product of the density of line primitives (DL) in a feature and the area of a line primitive.
- Another parameter used to create the features is the standard deviation for the rotation of the primitives. This parameter is unsuitable to include as an independent variable however, because it is only relevant for features containing at least two line and/or triangle primitives. This is because for circle primitives the rotation is irrelevant and if there is only one line or triangle primitive there is no relative rotation between line and/or triangle primitives. To ensure that the standard deviation for the rotation of primitives is irrelevant in these cases and to scale the influence of the rotation with the density of line and triangle primitives in the feature, the independent variable ROT is defined as in Equation (5.11):

$$ROT = \frac{\max(TT + TL - 1, 0)}{FA} * SD \quad (5.11)$$

In Equation (5.11) TT is the total number of triangles in a feature, TL is the total number of lines in a feature, FA is the area of the feature in μm^2 , and SD is the standard deviation for the rotation of the primitives in degrees.

- We define a variable CCD using the 100×100 discretization of the feature. We look at the pixels on the diagonal of the feature from the bottom left corner to the top right corner. Consider the pixels covered by a primitive as black pixels and the pixels not covered as white pixels. CCD is now defined as the number of times the color of the pixels changes going over the diagonal from the bottom left corner to the top right corner divided by the length of this diagonal in μm . The division by the length of the diagonal is made in order to make features of different sizes better comparable. Features consisting of a lot of separate primitives typically have a higher value for CCD , while features consisting of a small number of primitives or features for which the primitives are mostly clustered together typically have a lower value for this variable. Using this variable we try to investigate whether the extent to which the feature contains spread primitives influences the cellular response.
- The final group of variables is based on the two-dimensional discrete Fourier transform (DFT) of the discretization of the feature. The DFT represents the elements of the discretization of the feature as a sum of sinusoids with different wavenumbers. Eleven variables WN_x with values for x ranging from 0.1 to 4 are created. These variables represent the fraction of the total energy that is present in sinusoids with wavenumber approximately x . In Section 5.2.3 a detailed description of the DFT and the variables WN_x that are based on it, is given.

5.2.3 Discrete Fourier transform

This section describes the discrete Fourier transform (DFT) and how its results can be interpreted. Afterwards variables based on the DFT are defined, which are used in the regression analysis. The DFT transforms a signal (matrix of values) from the spatial to the frequency domain. To compute the DFT of a feature, we start with a discretization $f(j, k)$ of the feature. This is an $N \times N$ -matrix containing binary values. For features of size $10 \mu m$ by $10 \mu m$ N equals 100, for features of size $20 \mu m$ by $20 \mu m$ N equals 200, and for features of size $28 \mu m$ by $28 \mu m$ N equals 280. These values for N are chosen such that each element of f corresponds with a $0.1 \mu m$ by $0.1 \mu m$ area of the feature. Each element of f is 1 if the center of the area of the feature it corresponds with is within at least one of the primitives composing the feature, otherwise it is 0. For an example of a discretized feature, see Figure 5.2. Note that you can still recognize the individual triangle primitives, but the image is pixelated.

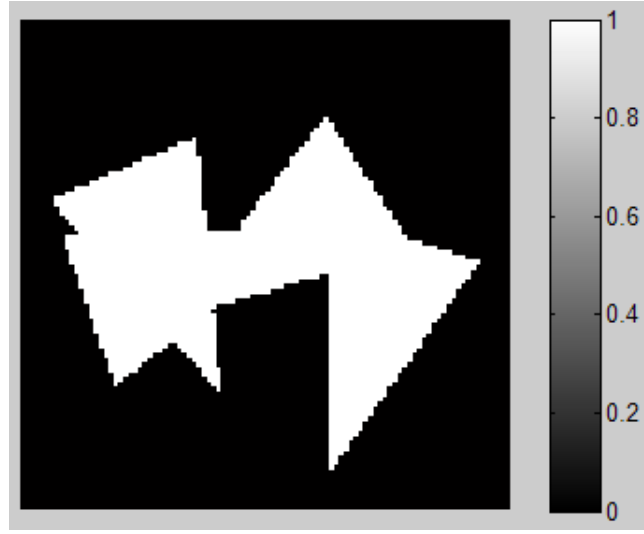


Figure 5.2: An example of a discretized feature.

In the following we define the wavenumber WN of a periodical wave as the number of times the wave has the same phase per unit of space. The wavenumber is the inverse of the wavelength λ . It can be seen as the spatial analog of frequency. As its unit we use μm^{-1} .

The DFT provides a matrix $F(p, q)$, which has size $N \times N$ as well. The components of this matrix are defined in terms of the matrix f as in Equation (5.12).

$$F(p, q) = \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} f(j, k) e^{-i(2\pi/N)pj} e^{-i(2\pi/N)qk} \quad (5.12)$$

$$p = 0, 1, \dots, N-1 \quad q = 0, 1, \dots, N-1$$

The inverse DFT is defined in Equation (5.13).

$$f(j, k) = \frac{1}{N^2} \sum_{p=0}^{N-1} \sum_{q=0}^{N-1} F(p, q) e^{i(2\pi/N)pj} e^{i(2\pi/N)qk} \quad (5.13)$$

$$j = 0, 1, \dots, N-1 \quad k = 0, 1, \dots, N-1$$

From Equation (5.13) it can be seen that the DFT represents the components of the matrix f as a sum of complex exponentials. To provide more insight, these complex exponentials can be rewritten as follows (see also [7]):

$$\begin{aligned} & e^{i(2\pi/N)pj} e^{i(2\pi/N)qk} \\ = & e^{i(2\pi/N)(pj+qk)} \end{aligned} \quad (5.14)$$

$$= e^{i(2\pi w/N)(\frac{pj}{w} + \frac{qk}{w})} \quad (5.15)$$

$$= e^{i(2\pi w/N)(\vec{r} \cdot \vec{n})} \quad (5.16)$$

In Equations (5.14) to (5.16) $w = \sqrt{p^2 + q^2}$. The unit vector in the direction (p, q) is given by $\vec{n} = (\frac{p}{w}, \frac{q}{w})$. The vector in the direction (j, k) in the spatial domain is given by $\vec{r} = (j, k)$.

For all points on a line perpendicular to \vec{n} in the spatial domain, the exponential $e^{i(2\pi w/N)(\vec{r} \cdot \vec{n})}$ takes the same value, since these points have the same projection $\vec{r} \cdot \vec{n}$ on \vec{n} . This means that in the spatial domain the exponential represents a sinusoid in the direction \vec{n} . The wavelength of this sinusoid is $\lambda = \frac{N}{w} \cdot 0.1 \mu m$ (the multiplication by $0.1 \mu m$ is made because the length of one unit in the spatial domain is chosen as $0.1 \mu m$, see earlier in this section). The wavenumber WN is equal to the inverse of the wavelength: $WN = \frac{10w}{N} \mu m^{-1}$.

As an example, consider the 40×40 Fourier transform F with all elements equal to 0, except for $F(0, 0)$ and $F(4, 4)$, which are both equal to 200. One unit in the space domain is equal to $0.1 \mu m$. Element $F(0, 0)$ is the constant component of the Fourier transform. From Equation (5.12) it can be seen that $F(0, 0)$ is equal to the sum of all the $f(j, k)$. Element $F(4, 4)$ corresponds with a sinusoid with wavenumber $WN = \frac{10w}{N} = \frac{10\sqrt{p^2+q^2}}{N} = \frac{10\sqrt{4^2+4^2}}{40} = \sqrt{2} \approx 1.41 \mu m^{-1}$ in the direction $\vec{n} = (\frac{p}{w}, \frac{q}{w}) = (\frac{4}{\sqrt{4^2+4^2}}, \frac{4}{\sqrt{4^2+4^2}}) = (\frac{1}{2}\sqrt{2}, \frac{1}{2}\sqrt{2})$. Applying the inverse DFT to F provides the matrix f representing the signal in the space domain. A plot of the absolute value of the elements of f is shown in Figure 5.3. Note the presence of a sinusoid with wavenumber $\sqrt{2} \mu m^{-1}$ in the direction $(\frac{1}{2}\sqrt{2}, \frac{1}{2}\sqrt{2})$.

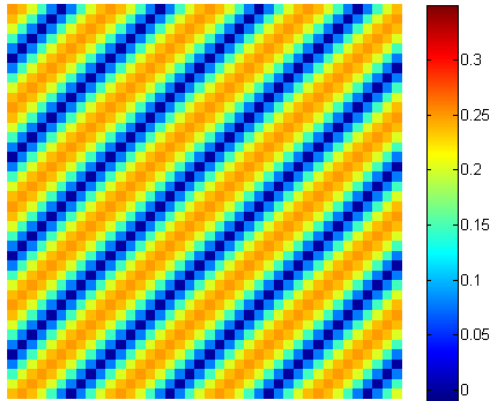


Figure 5.3: A graphical representation of the absolute value of the elements of the matrix f obtained after applying the inverse DFT to a 40×40 matrix F with all elements equal to 0, except for $F(0, 0)$ and $F(4, 4)$, which are both equal to 200.

Parseval's theorem is a well-known result that relates the total energy contained in

a signal in the space (or time) domain to the total energy contained in the signal after applying the Fourier transform. In case the type of the Fourier transform used is the DFT and the signal has size $N \times N$, Parseval's theorem is given by Equation (5.17).

$$\sum_{j=0}^{N-1} \sum_{k=0}^{N-1} |f(j, k)|^2 = \frac{1}{N^2} \sum_{p=0}^{N-1} \sum_{q=0}^{N-1} |F(p, q)|^2 \quad (5.17)$$

The summation on the left hand side of Equation (5.17) can be interpreted as the total energy contained in the signal in the space domain and the summation on the right hand side can be interpreted as the total energy contained in the signal after applying the DFT. Note that the left side of the equation is equal to the number of elements of the discretization f that are 1, i.e. the total number of pixels of the discretization covered by primitives contained in the feature.

We now create variables WN_x which represent the fraction of the total energy contained in the signal after applying the DFT, that is present in sinusoids with wavenumber approximately x . We include the sinusoids that have a wavenumber WN that differs less than 10 percent from x , that is $0.9 x < WN < 1.1 x$. At first glance it appears these are just the sinusoids associated with the elements $F(p, q)$ for which $0.9 x < \frac{10w}{N} = \frac{10\sqrt{p^2+q^2}}{N} < 1.1 x$. However, because the value of the exponentials $e^{i(2\pi/N)(pj+qk)}$ is only sampled at integer values for j and k , something interesting happens. For a one-dimensional example of this phenomenon, see the plot of the functions $f(x) = \cos(\frac{1}{6} \cdot 2\pi x)$ and $g(x) = \cos(\frac{5}{6} \cdot 2\pi x)$ in Figure 5.4. Even though these functions have very different wavenumbers, at integer values for x both f and g take the same value. Something similar happens when we rewrite the exponential belonging to the element $F(p, N - q)$.

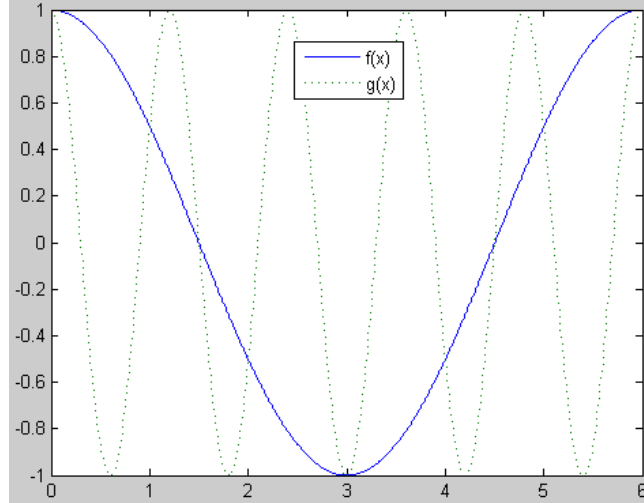


Figure 5.4: A plot of the functions $f(x) = \cos(\frac{1}{6} \cdot 2\pi x)$ and $g(x) = \cos(\frac{5}{6} \cdot 2\pi x)$. Note that for integer values of x , f and g take the same value.

$$e^{i(2\pi/N)pj} e^{i(2\pi/N)(N-q)k} = e^{i(2\pi/N)(pj+(N-q)k)} \quad (5.18)$$

$$= e^{i(2\pi Nk/N)} e^{i(2\pi/N)(pj-qk)} \quad (5.19)$$

$$= e^{i(2\pi/N)(pj-qk)} \quad (5.20)$$

$$= e^{i(2\pi w/N)(\frac{pj}{w} + \frac{-qk}{w})} \quad (5.21)$$

$$= e^{i(2\pi w/N)(\vec{r} \cdot \vec{n})} \quad (5.22)$$

In Equations (5.18) to (5.22) $w = \sqrt{p^2 + q^2}$ and $\vec{r} = (j, k)$ as before. The vector \vec{n} is the unit vector in the direction $(\frac{p}{w}, \frac{-q}{w})$. Note that because k only takes integer values, the factor $e^{i(2\pi Nk/N)}$ in Equation (5.19) is equal to 1. The exponential in Equation (5.22) again represents a sinusoid with wavenumber $\frac{10w}{N}$, but this time in another direction. From Equations (5.18) to (5.22) it can be seen that $e^{i(2\pi/N)pj} e^{i(2\pi/N)(N-q)k}$ takes the same values as $e^{i(2\pi w/N)(\vec{r} \cdot \vec{n})}$ when j and k only take integer values. Since these exponentials are indistinguishable in our case where we only sample for integer values for j and k , we also consider the sinusoid associated with $F(p, N-q)$ as having wavenumber $WN = \frac{10\sqrt{p^2+q^2}}{N} = \frac{10w}{N}$.

Similarly, it can be shown that when we only sample for integer values for j and k , the sinusoid associated with $F(N-p, q)$ is indistinguishable from a sinusoid with wavenumber $WN = \frac{10w}{N}$ in the direction $(\frac{-p}{w}, \frac{q}{w})$ and the sinusoid associated with $F(N-p, N-q)$ is indistinguishable from a sinusoid with wavenumber $WN = \frac{10w}{N}$ in the direction $(\frac{-p}{w}, \frac{-q}{w})$. Because of this, we consider the sinusoids associated with $F(p, q)$, $F(p, N-q)$, $F(N-p, q)$, and $F(N-p, N-q)$ as having wavenumber $WN = \frac{10\sqrt{p^2+q^2}}{N} = \frac{10w}{N}$.

The variables WN_x can now be defined as in Equation (5.23).

$$WN_x = \frac{\sum_{(p,q) \in S} |F(p, q)|^2}{\sum_{u=0}^{N-1} \sum_{v=0}^{N-1} |F(u, v)|^2} \quad (5.23)$$

with $S = \{(p, q) \mid p \in \{0, 1, \dots, N-1\},$

$q \in \{0, 1, \dots, N-1\},$

$$0.9x < \frac{10\sqrt{p^2+q^2}}{N} < 1.1x \vee$$

$$0.9x < \frac{10\sqrt{p^2+(N-q)^2}}{N} < 1.1x \vee$$

$$0.9x < \frac{10\sqrt{(N-p)^2+q^2}}{N} < 1.1x \vee$$

$$0.9x < \frac{10\sqrt{(N-p)^2+(N-q)^2}}{N} < 1.1x \}$$

The numerator of the fraction on the right hand side of Equation (5.23) represents the energy contained in sinusoids with a wavenumber that differs less than 10% from x . The denominator represents the total energy contained in the DFT of the signal. For a graphical representation of which elements of F are associated with sinusoids with wavenumber approximately 2, see Figure 5.5. Note the symmetry that is present. The elements of F close to the corners are associated with sinusoids with low wavenumbers, while the elements of F far from the corners are associated with sinusoids with high wavenumbers.

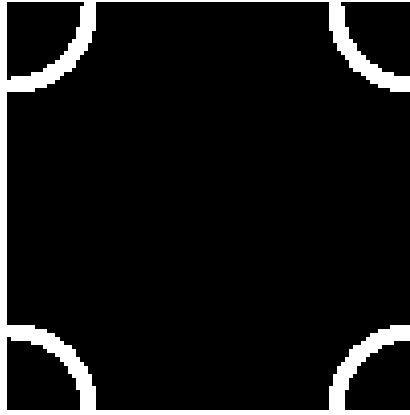


Figure 5.5: A graphical representation of the elements of the matrix F obtained after applying the DFT which are included when computing the energy contained in sinusoids with a wavenumber that differs less than 10 % from 2. The elements colored white are included. In this case the feature is approximated by a 100×100 discretization.

Eleven variables WN_x are created representing different wavenumbers x . The variables that are created are: $WN_{0.1}$, $WN_{0.2}$, $WN_{0.3}$, $WN_{0.4}$, $WN_{0.5}$, $WN_{0.7}$, WN_1 , $WN_{1.5}$, WN_2 , WN_3 , and WN_4 . The smallest wavenumber which presence is measurable is given by the minimum of $WN = \frac{10w}{N}$. Since the wavenumber has to be measurable for all values of N , we take $N = 100$ since in this case WN is biggest. The minimum is attained for $w = 1$ (the trivial case $w = 0$ is excluded, because as can be seen from Equation (5.12), $F(0,0) = \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} f(j,k)$, which is strongly related to the variable FCP created in Section 5.2.2) which gives a minimum measurable wavenumber of 0.1. The energy contained in sinusoids with a wavenumber larger than 5 is not measurable, since this includes elements of the matrix F which are not in the top left quadrant. The sinusoids corresponding with these elements are indistinguishable from sinusoids with wavenumbers smaller than 5, as was shown in Equations (5.18) to (5.22). The largest wavenumber considered is 4, since wavenumbers close to 5 are undesirable. This is because of the 10% margin, which causes elements of F not in the top left quadrant to be included if x is chosen to close to 5. For these reasons the values for x are chosen between 0.1 and 4.

For a typical example of a feature that has a high value for $WN_{0.1}$ and a typical example of a feature that has a high value for WN_4 , see Figure 5.6. Features that have high values for $WN_{0.1}$ often have dimensions $10 \mu m$ by $10 \mu m$ and often contain (mostly circle and line) primitives clustered in the middle. This is probably because the features are designed in such a way that $10 \mu m$ by $10 \mu m$ features with primitives clustered in the middle are relatively common, and because sinusoids that have a wavenumber of 0.1 (and thus have wavelength $10 \mu m$) are very suitable for describing such a feature. These sinusoids can be translated such that their maximum overlaps with the cluster of primitives in the middle, while their minimum is attained at the edges of the feature, where there are no primitives. The feature with dimensions $20 \mu m$ by $20 \mu m$ with the highest value for $WN_{0.1}$ can be seen in Figure 5.7. In this feature one can recognize the presence of sinusoids with wavenumber 0.1 in the vertical direction. Features that have high values for WN_4 typically contain primarily small triangle primitives that are well spread over the feature.

5.3 Stepwise regression

Stepwise regression is a method to select a number of independent variables for a regression model from a collection of possible independent variables. The reason that stepwise regression is used instead of including all of the possible independent variables in the model, is that we do not want to unnecessarily complicate the model by including insignificant variables.

Before the stepwise regression algorithm can be used, we have to choose values for parameters p_{enter} and p_{remove} . The parameter p_{enter} represents the probability that a variable x_i not currently in the regression model is added to the model while it is actually insignificant ($\beta_i = 0$), controlling for the variables in the model. The parameter p_{remove} represents the probability that a variable x_i currently in the regression model is kept in the model while it is actually insignificant ($\beta_i = 0$), controlling for the variables in the model. For a more formal description of p_{enter} and p_{remove} , consider model 1 with a certain set of independent variables and another model 2 with the same set of independent variables plus one other variable x . We now define F as in Equation (5.24).

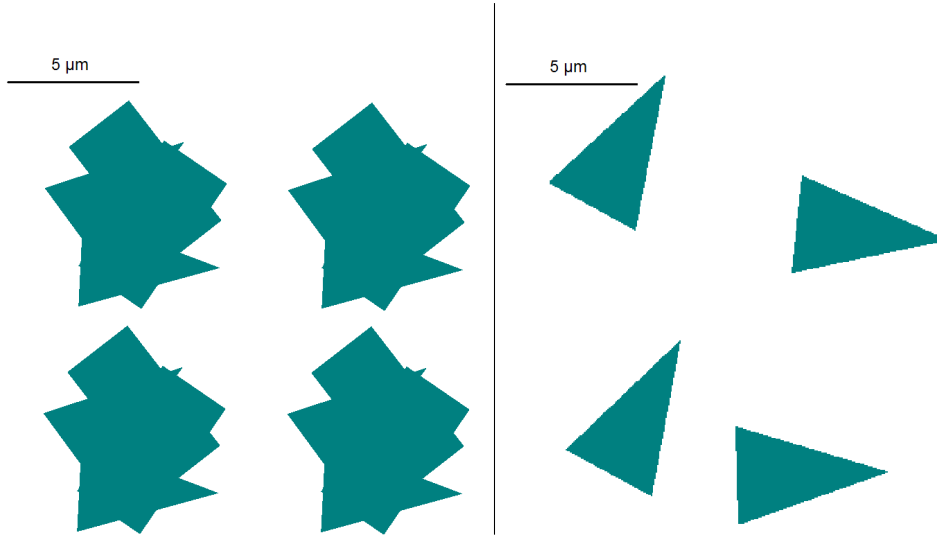


Figure 5.6: The feature on the left is a typical feature with a high value for $WN_{0.1}$ (it contains relatively much energy in sinusoids with a low wavenumber). The feature on the right is a typical feature with a high value for WN_4 (it contains relatively much energy in sinusoids with a high wavenumber).

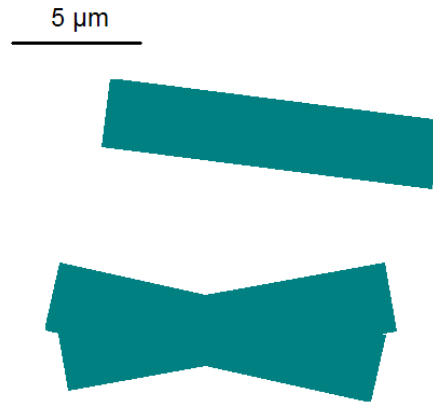


Figure 5.7: The feature with dimensions $20 \mu m$ by $20 \mu m$ with the highest value for $WN_{0.1}$. Note that this feature can be approximated quite well with sinusoids in the vertical direction with wavenumber 0.1, which means that they take their maximum twice on a length of $20 \mu m$.

$$F = \frac{(N - (p_2 + 1))(RSS_{err,1} - RSS_{err,2})}{RSS_{err,2}} \quad (5.24)$$

In Equation (5.24) N is the size of the data set, p_2 is the number of variables in model 2, and $RSS_{err,i} = \sum_{j=1}^N (Y_j - \hat{Y}_{i,j})^2$ is the part of the variance of Y unexplained by model j . $\hat{Y}_{i,j}$ is the predicted value for Y_j in model i . Assuming that there is no relation between Y and x ($\beta_x = 0$), F is distributed as an F -distribution with $(1, N - p_2 + 1)$ degrees of freedom. When deciding whether to add a variable to a model, we add it only if we have to reject the null hypothesis $\beta_x = 0$ at a significance level of p_{enter} , that is F takes a value larger than the critical point of the F -distribution with $(1, N - p_2 + 1)$ degrees of freedom with rejection probability p_{enter} . Similarly, we remove a variable x from a model if we do not have to reject the null hypothesis $\beta_x = 0$ at a significance level of p_{remove} .

Note that p_{remove} has to be at least as big as p_{enter} , because otherwise a variable just added to the model can sometimes be removed again immediately, creating a cyclic behavior of adding and removing the same variable over and over again. In choosing values for p_{enter} and p_{remove} a balance has to be found. Choosing the values higher results in an increased chance for irrelevant variables to be included in the model, but choosing them lower results in an increased chance for relevant variables not to be included in the model. Typical values chosen are $p_{enter} = 0.05$ and $p_{remove} = 0.10$.

Stepwise regression starts with a model without any independent variables, but with an intercept β_0 . The regression equation is as in Equation (5.25):

$$Y = \beta_0 \quad (5.25)$$

Let S denote the set of independent variables in the regression model at a certain point. The stepwise regression algorithm proceeds as follows:

- **Step 1:** Let the regression equation at this point be given by Equation (5.26):

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k \quad (5.26)$$

For each of the variables x_i in Equation (5.26), consider the null hypothesis $H_0 : \beta_i = 0$. If we cannot reject the null hypothesis for the chosen significance level p_{remove} , variable x_i is considered insignificant in a model containing the variables $S \setminus x_i$. Variable x_i is removed from S and new values for the β_i are computed (the β_i usually change when variables are added to or removed from the model). In case there are multiple insignificant variables, the least significant one is removed. If a variable is removed during this step, step 1 is repeated. Otherwise, the stepwise regression algorithm proceeds to step 2.

- **Step 2:** For each variable \tilde{x} not in S , consider the model with independent variables $S \cup \{\tilde{x}\}$. The regression equation is given by Equation (5.27):

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_1 + \dots + \beta_k x_k + \tilde{\beta} \tilde{x} \quad (5.27)$$

Consider the null hypothesis $H_0 : \tilde{\beta} = 0$. If we have to reject the null hypothesis for the chosen significance level p_{enter} , \tilde{x} is considered significant in the model with the variables contained in S as independent variables. Variable \tilde{x} is added

to S and new values for the β_i are computed. In case there are multiple significant variables not included in S , the most significant one is added. If a variable is added to S , the algorithm goes back to step 1. Otherwise, the algorithm stops.

In some cases it may be necessary to terminate the algorithm, for example when it gets in a loop returning to the same collection of variables after a series of additions and removals of variables. In addition a limited number of iterations can be imposed, to limit the running time of the algorithm. If the algorithm terminates normally, because of its construction, it only terminates with a set of independent variables S that does not contain any insignificant variables. In addition, variables that are not included in S are not significant if added to the model with the variables in S as independent variables. In this sense, S can be considered as a local optimum.

Chapter 6

Results

In this chapter we apply the stepwise regression algorithm and provide the results. Afterwards, the main results are summarized.

6.1 Results of the regression analysis

In this section we perform the regression analysis. This analysis is first performed for the dependent variable without any added noise. This analysis is described in detail. Afterwards we perform the analysis for the dependent variable with noise added. This analysis is similar to the analysis for the case with no noise. It is described in less detail.

6.1.1 Dependent variable without noise

We first perform the regression analysis with *LOGFCP* as the dependent variable. As the independent variables we use the variables listed in Appendix B. The feature number (*NUM*), the row number (*ROW*), and the column number (*COL*) are expected to have no influence on the independent variable, since all features are created randomly without using information about the feature, row or column number. These variables are included however, to see whether the regression analysis wrongly identifies these variables as relevant.

We apply the stepwise regression algorithm described in Section 5.3 with parameters $p_{enter} = 0.05$ and $p_{remove} = 0.10$. Before we apply the algorithm, we split the data set in two. There are 2176 features in total. The first 1088 features are included in the first set and the second 1088 features are included in the second set. The reason to split the data set in two is to get a better idea about the reliability of the model. In stepwise regression analysis, there is always the possibility that relevant variables are omitted from the model, or that irrelevant variables are included (see also Section 5.3). We fit a model for the first set and check whether this model is appropriate for the second set as well.

The stepwise regression analysis is performed using SPSS. Applying the stepwise regression algorithm to the first half of the data set provides the regression equation given by Equation (6.1).

$$\begin{aligned}
LOGFCP = & -1.21 - 0.00174 \cdot ROW + 12.7 \cdot DC + 19.1 \cdot DT \\
& + 12.4 \cdot DL + 1.10 \cdot CA + 1.15 \cdot TA \\
& + 1.40 \cdot LA - 0.0284 \cdot ROT + 1.72 \cdot CCD \\
& - 0.479 \cdot WN_{0.1} - 19.4 \cdot WN_1 - 92.6 \cdot WN_{1.5}
\end{aligned} \tag{6.1}$$

The coefficient of determination R^2 is a measure that describes how much of the variance of the dependent variable is explained by the regression model. R^2 is defined as in Equation (6.2).

$$R^2 = \frac{SS_{reg}}{SS_{tot}} = 1 - \frac{SS_{err}}{SS_{tot}} \tag{6.2}$$

In Equation (6.2) $SS_{tot} = \sum_{i=1}^N (FCPLOG_i - \overline{FCPLOG})^2$ is the total sum of squares (total variance of the dependent variable). $SS_{reg} = \sum_{i=1}^N (FCPLOG_i - \widehat{FCPLOG})^2$ is the sum of squares due to regression (explained variance by the regression model). $SS_{err} = \sum_{i=1}^N (FCPLOG_i - \widehat{FCPLOG}_i)^2$ is the error sum of squares (part of the variance that is unexplained). R^2 takes values between 0 and 1. The closer to 1 the value of R^2 is, the more of the variance is explained.

For the final model obtained after applying the stepwise regression algorithm to the first part of the data set, $R^2 = 0.860$.

The order in which variables are entered in and removed from the model can be seen in Figure 6.1 (in this case no variables are removed).

Variables Entered/Removed^{a,b}

Model	Variables Entered	Variables Removed
1	WN1.5	.
2	CCD	.
3	LA	.
4	DT	.
5	CA	.
6	TA	.
7	DC	.
8	DL	.
9	WN0.1	.
10	ROT	.
11	WN1	.
12	ROW	.

a. First half of the data set

b. Dependent Variable: FCPLOG

Figure 6.1: The order in which the variables are added to the model for $FCPLOG$ during the application of the stepwise regression algorithm to the first half of the data set.

In Figure 6.2 information is given on the coefficients in the regression equation. The column labeled 'Beta' gives the values of the coefficients β_i . The column labeled 'B' gives the values of standardized coefficients. The standardized coefficients are obtained by standardizing the variables before applying the stepwise regression algorithm. Standardization is done by taking the variable value, subtracting the mean of the variable and afterwards dividing by the standard deviation of the variable. This process makes coefficients for variables which are measured in different units better comparable. The column labeled 'Sig.' gives the significance level of the variable. The lower the value, the more significant the variable is. Variables which are significant at a level larger than 0.1 are removed from the model. For a more detailed description of the significance level, see Section 5.3.

Coefficients ^{a,b}				
Model		Beta	B	Sig.
1	(Constant)	.069		.097
	WN1.5	-155.302	-.544	.000
12	(Constant)	-1.209		.000
	WN1.5	-92.577	-.325	.000
	CCD	1.719	.217	.000
	LA	1.399	.602	.000
	DT	19.095	.349	.000
	CA	1.099	.487	.000
	TA	1.148	.393	.000
	DC	12.722	.259	.000
	DL	12.432	.236	.000
	WN0.1	-.479	-.102	.000
	ROT	-.028	-.060	.000
	WN1	-19.441	-.096	.002
	ROW	-.002	-.029	.013

a. First half of the data set

b. Dependent Variable: FCPLOG

Figure 6.2: Information on the coefficients in the regression equation for *FCPLOG* during the application of the stepwise regression algorithm to the first half of the data set. This information is given for the first and the final model. The column labeled 'Beta' lists the values of the coefficients β_i and the column labeled 'B' lists the values of the standardized coefficients. The column labeled 'Sig.' lists the significance level of each variable.

In Figure 6.3 information is given on the variables not in the regression equation. The column labeled 'Beta In' gives the standardized coefficient for each variable if it is included as the next variable in the model. The column labeled 'Sig.' gives the significance level for each variable. Variables which are significant at a level smaller than 0.05 are candidates for inclusion in the model in the next step (the most significant variable is included first).

From the regression analysis, the following can be observed:

Excluded Variables

Model		Beta In	Sig.
1	NUM	-.043	.088
	ROW	-.044	.081
	COL	.033	.197
	WN0.1	.273	.000
	WN0.2	-.073	.016
	WN0.3	.195	.000
	WN0.4	.159	.000
	WN0.5	-.290	.000
	WN0.7	.096	.036
	WN1	.021	.734
	WN2	.391	.002
	WN3	.144	.314
	WN4	.360	.003
	DC	.115	.000
	DT	.287	.000
	DL	.333	.000
	TA	.162	.000
	CA	.156	.000
	LA	.356	.000
	ROT	.313	.000
	CCD	.460	.000
12	NUM	.210	.580
	COL	.007	.572
	WN0.2	.005	.761
	WN0.3	.004	.811
	WN0.4	.031	.130
	WN0.5	-.012	.574
	WN0.7	.018	.565
	WN2	.025	.685
	WN3	-.042	.550
	WN4	-.118	.073

Figure 6.3: Information on the variables not in the regression equation for *FCPLOG* during the application of the stepwise regression algorithm to the first half of the data set. This information is given for the first and the final model. The column labeled ‘Beta In’ gives the standardized coefficient for each variable if it is included as the next variable in the model. The column labeled ‘Sig.’ gives the significance level for each variable.

- The density and area of circle, triangle, and line primitives all appear with positive coefficients in the regression equation. From Figure 6.2 it can be seen that they are all highly significant in the final model. This is as expected since more and larger primitives typically lead to a higher fraction of the feature being covered by primitives.
- The row number appears in the regression equation, while the features are designed in such a way that there should be no relation between the row number and the fraction of the feature covered by primitives. This can be explained by the value of 0.05 for p_{enter} . This value can be interpreted as the probability of a variable being included in the regression model, while it is actually irrelevant. Since there are three variables that should be irrelevant for the value of FCP (NUM , ROW , and COL), it is not very unlikely that one is wrongly included in the regression equation. When testing the model on the second half of the data set, it turns out that ROW is indeed irrelevant for the value of FCP for this data set.
- From Figure 6.3 it can be seen that if ROT is added to the first model (with only a constant term and $WN_{1.5}$ as a variable), it is included with a positive coefficient. In the final model ROT is included with a negative coefficient. This can be explained because in the computation of ROT , the standard deviation of the rotation is scaled by the number of triangle and line primitives. Since more triangle and line primitives typically lead to a higher FCP , it is logical that ROT initially has a positive coefficient. In the final model however, the density of triangle and line primitives is already included in the model. Since the influence of the density is already included, ROT can be interpreted as a compensational effect representing the additional effect of the amount of rotation of primitives.
- From Figure 6.3 it can be seen that most of the variables WN_x are highly significant when added to the first model. In the final model, only three of the variables WN_x are included. Apparently, much of the information contained in the variables WN_x is also contained in other variables. Adding more of the variables WN_x to the final model does not give much more information about FCP .

The largest difference between the actual value of $FCPLOG$ and the value predicted by the regression equation occurs for feature number 685. The actual value is 0.0949, while the predicted value is -0.656. This corresponds with values for FCP of 0.524 and 0.342 respectively. This feature is shown in Figure 6.4. A possible explanation for the fact that the predicted value is much too low is that considering the fact that the primitives are pretty large, they overlap relatively little. In addition the CCD for the feature is low (if the other diagonal is used to calculate the CCD , the value is larger).

We now use the second half of the data set in order to determine the quality of the model obtained with the regression analysis. First we use the model for the first half of the data set, given by Equation (6.1), to predict values for $FCPLOG$ for the features in the second half of the data set. If the model based on the first half of the data set is suitable for predicting values for the second half of the data set, this suggests that the model is valid for all features created similarly to the first generation of features.

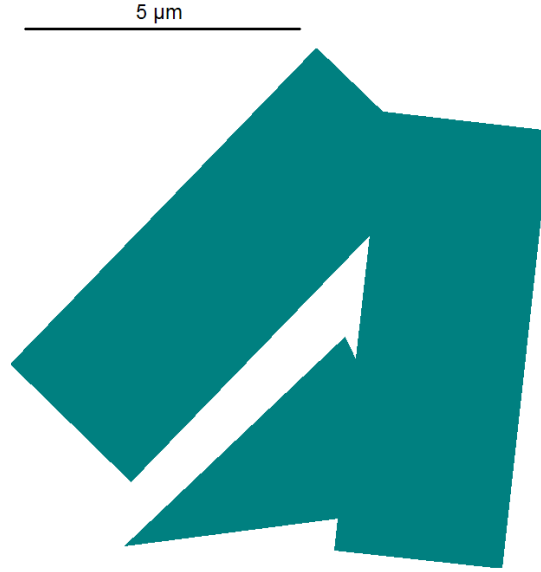


Figure 6.4: The feature in the first half of the data set for which the predicted and actual value for *LOGFCP* differ most.

Computing R^2 for the model based on the first half of the data set applied to the second half of the data set, gives a value of $R^2 = 0.854$. When the stepwise regression algorithm is applied to the second half of the data set, the final model has a value of $R^2 = 0.866$. In this case it turns out that the model based on the first half of the data set is not much worse in predicting values for *FCPLOG* for the second half of the data set than the model that is based on the second data set itself. This suggests that the model based on the first half of the data set is suitable for describing the dependence of *FCPLOG* on the dependent variables for features created similarly to the features in the first generation.

Next we test whether the variables in the final model found for the first half of the data set are significant in the model for the second half of the data set as well. If this is not the case, it is likely that these variables are included in the model for the first half of the data set, because by chance there is some relation between the variables and the dependent variable. If this relation is not present in the second data set, this is an indication that in general there is no (or only a weak) relation between the variable and the dependent variable. We then choose not to include the variable in the final model for the complete data set.

We determine the optimal regression model for the second half of the data set with all the variables in the final model for the first half of the data set included. The value of R^2 for this model is $R^2 = 0.863$. In this model *ROW* has a significance level of 0.775. Since $p_{remove} = 0.10$, the variable *ROW* is not considered significant in the model and is removed. In the new regression model, *ROT* has a significance level of 0.110. Since this is larger than p_{remove} as well, *ROT* is removed from the model. In the new model, all variables are significant at a level less than 0.001, so there is no need to remove any more variables. The value of R^2 is still $R^2 = 0.863$, so removal of the variables *ROW* and *ROT* does not (significantly) affect the quality of the model.

From this analysis it appears that the inclusion of *ROW* in the model for the first half of the data set is probably because there is some relation between *ROW* and *FCPLOG* in this data set by chance. For the second half of the data set there are

no signs of any relation between *ROW* and *FCPLOG*. The variable *ROT* is close to being significant enough in the model for the second half of the data set, but is removed from the model anyway. Since it is significant in the first model and almost significant in the second model, it is probable that there is some relation between *ROT* and *FCPLOG*. However, because keeping *ROT* in the model hardly increases R^2 , there is no need to have *ROT* in the model. To keep the model simple, it is recommended to remove *ROT* from the model.

We formulate a final regression model for the complete data set. As the independent variables we use the variables that are significant for the first half of the data set except for *ROW* and *ROT*, which are found insignificant in the model for the second half of the data set. The final model is as in Equation (6.3).

$$\begin{aligned}
LOGFCP = & -1.217 + 13.6 \cdot DC + 18.9 \cdot DT + 11.8 \cdot DL \\
& + 1.09 \cdot CA + 1.10 \cdot TA + 1.36 \cdot LA \\
& + 1.86 \cdot CCD - 0.431 \cdot WN_{0.1} - 25.3 \cdot WN_1 \\
& - 97.3 \cdot WN_{1.5}
\end{aligned} \tag{6.3}$$

The value of R^2 for this model is $R^2 = 0.860$.

6.1.2 Dependent variable with noise

Noise level 0.1

We now perform the regression analysis using the dependent variable with noise added $LOGFCPN_{0.1}$ as described in Section 5.2.1. We use the same independent variables as in the case without noise and we still use $p_{enter} = 0.05$ and $p_{remove} = 0.1$.

For the first half of the data set, the nine variables included in the final model are *DC*, *DT*, *DL*, *CA*, *TA*, *LA*, *CCD*, $WN_{0.1}$ and $WN_{1.5}$. Note that these are the same variables as the variables in the final regression model for the case with no noise given by Equation (6.3), except that WN_1 is not included. The value of $R^2 = 0.455$ is much lower than the value of $R^2 = 0.860$ in the situation without noise. This is as expected, since the contribution of the noise term is impossible to predict.

Next we apply the model found for the first half of the data set to the second half of the data set. This gives a value of $R^2 = 0.480$. Running the regression algorithm again on the second half of the data set produces a model with a value of $R^2 = 0.496$. The model for the first half of the data set is not much worse at predicting the value of the dependent variable for the second half of the data set than a model built using the data of the second half of the data set itself. This indicates that even though it is difficult to predict the value of the dependent variable because of the noise, a model based on one data set seems to be suitable for describing the relation between the dependent and independent variables for another data set of features which are created similarly to the first generation of features.

Next we create a regression model for the second half of the data set, using only the variables included in the final model for the first half of the data set. In the model for the second half of the data set, all of these variables are significant. The least significant variable is $WN_{0.1}$, which has a significance level of 0.033.

The model based on the first half of the data set is adequate for predicting the dependent variable for the second half of the data set. All variables in the model for the first half of the data set are significant in the model for the second half of the

data set as well. For these reasons we see no need to remove or add any variables and base the final regression model for the complete data set on the variables in the final model for the first half of the data set. The regression model for the complete data set is given by Equation (6.4).

$$\begin{aligned}
LOGFCPN_{0.1} = & -1.301 + 15.6 \cdot DC + 22.8 \cdot DT + 15.4 \cdot DL \\
& + 1.21 \cdot CA + 1.19 \cdot TA + 1.50 \cdot LA \\
& + 1.92 \cdot CCD - 0.625 \cdot WN_{0.1} - 147 \cdot WN_{1.5}
\end{aligned} \tag{6.4}$$

This model has a value of $R^2 = 0.470$. All variables included in the final model are very significant. The least significant variable is $WN_{0.1}$ with a significance level of 0.002.

Noise level 0.3

Next we consider the dependent $FCPLOGN_{0.3}$ which includes even more noise. Again we use the same independent variables and values of $p_{enter} = 0.05$ and $p_{remove} = 0.1$.

The final regression model for the first half of the data contains the variables DT , CA , TA , LA , CCD . These variables again form a subset of the variables included in the model without noise and with a noise level of 0.1. The value for R^2 for this model is $R^2 = 0.089$. Increasing the noise has drastically decreased the value of R^2 . The least significant variable in the final model is DT , which has a significance level of 0.004. The variable not in the final model which is most significant when added to the model, is DL with a significance level of 0.082.

Again we apply the model found for the first half of the data set to the second half of the data set. This gives a value of $R^2 = 0.113$. A new regression model for the second half of the data set has a value of $R^2 = 0.135$. In this case the new model for the second half of the data set performs significantly better than the model for the first half of the data set at predicting the value of the dependent variable for the second half of the data set. However, the model for the first half of the data set still does a reasonably good job.

Next we make a regression model for the second half of the data set using only the five variables included in the final model for the first half of the data set. In this model, DT is not significant. It has a significance level of 0.930 and is removed from the model. In the new model with four variables, CCD is not significant either with a significance level of 0.159. After removing this variable we obtain the final model containing only the variables CA , TA , and LA . All of these variables are significant at a level less than 0.001.

Since the variables DT and CCD are not significant in the model for the second data set, we choose not to include them in the model for the complete data set. The only variables that are included are CA , TA , and LA . The model for the complete data set is given by Equation (6.5).

$$LOGFCPN_{0.3} = -2.192 + 1.21 \cdot CA + 1.19 \cdot TA + 1.50 \cdot LA \tag{6.5}$$

This model has a value of $R^2 = 0.100$. All variables in the model are significant at a level less than 0.001.

The analysis for the case with noise level 0.3 suggests that CA , TA , and LA have a strong influence on FCP . To confirm this, we make a regression model for the case with no noise using only these three variables. The model found has a value of $R^2 = 0.765$. The final model obtained for the case with no noise with all independent variables considered (see Section 6.1.1) has a value of $R^2 = 0.860$. The seven additional variables that are included in this model only increase R^2 by 0.095. This means a large part of the variance of FCP can be explained by the values of CA , TA , and LA alone.

6.2 Summary of the results

In both the case without noise and the cases with noise, the stepwise regression algorithm succeeds in identifying variables that significantly influence the dependent variable FCP . The set of variables in the model for the case with no noise forms a superset of the set of variables in the model for the case with noise level 0.1. This set itself forms a superset of the set of variables in the model for the case with noise level 0.3. This suggests that the variables that describe the dependent variable best remain in the model even when a lot of noise is added, while the influence of variables that have a weaker relation with the dependent variable is overshadowed by the noise.

The variables CA , TA , and LA representing the area of the primitives explain a large part of the variance of FCP . Even when much noise is present, these variables remain significant (a noise level of 0.3 is very high, since this noise level means that more than 4% of the time the noise is larger than 0.6 (two standard deviations), changing a value of $FCP = 0.2$ to a measured value of $FCP = 0.8$, for example). When less noise is present other variables representing the density of the primitives (DC , DT , and DL), the number of color changes over the diagonal (CCD), and the amount of energy contained in parts of the signal with certain wavenumber ($WN_{0.1}$, WN_1 , and $WN_{1.5}$) become significant.

A regression model obtained using part of the data set is reasonably suitable for another part of the data set. Using the stepwise regression algorithm to obtain a new regression model specifically for that part of the data set, does not provide a much better model. This suggests that a model obtained using the complete data set is suitable for a new set of features which are created similarly to the first generation of features.

Chapter 7

Local search algorithms for improvement of features

In this chapter local search algorithms that can be used to search for higher quality features are discussed. These algorithms are not worked out in detail, but the ideas presented in this chapter can be used as guidance when an algorithm for feature improvement is implemented at a later stage.

Before new features can be created, a decision has to be made on whether the new features are built using circle, triangle, and line primitives, like the first generation of features. The alternative is to drop this restriction and allow a bigger class of possible features. Both choices have advantages and disadvantages. The advantages of keeping the restriction of creating new features using primitives, include:

- It is easier to handle the technical limitations mentioned in Section 2.2.3. These limitations prescribe a minimum ‘diameter’ for feature elements of $3\ \mu m$. As long as the primitives have a minimum size, the elements they form by overlapping are at least as big. If the features are created in another way, it is harder to control the size of individual elements of the feature.
- Some of the variables used in the regression analysis (see Section 5.2.2) are only defined for features built with circle, triangle, and line primitives. When other types of features are allowed, these variables cannot be used anymore.

The main disadvantage is that most features are impossible to build using only circle, triangle, and line primitives. For some features, it is even hard to approximate them using a limited number of circle, triangle, and line primitives. See Figure 7.1 for an example of such a feature.

An obvious approach for searching for high quality features is to use a feature that is known to be of high quality as a basis and to slightly change it. Hopefully an even higher quality feature is obtained in this way. Since this approach fits in the framework of local search algorithms, we present ideas on local search algorithms in the rest of this chapter. We discuss some of the opportunities and difficulties present when using local search algorithms to search for higher quality features.

In summary, local search algorithms work as follows. They start with a start solution. Next they move to a neighbor solution, that is a solution that is close to the current solution in a certain sense. The neighbor solution can be selected by determining a promising search direction (by using information on the quality of solutions in the neighborhood) and moving in that direction. An alternative is to randomly select a neighbor and if this neighbor is of inferior quality, to only move to

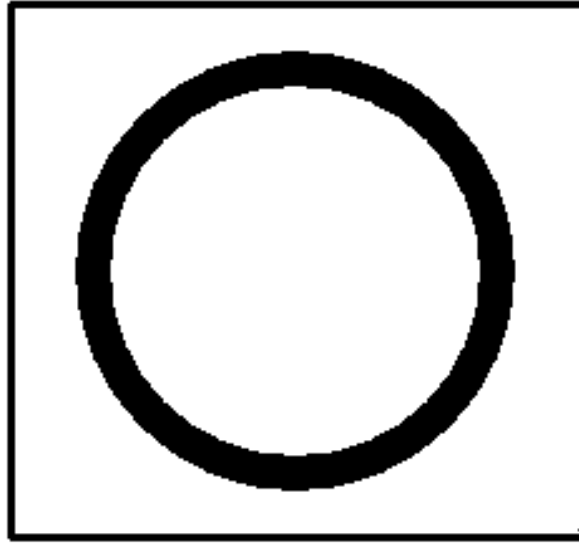


Figure 7.1: An example of a feature that is difficult to approximate using a limited number of circle, triangle, and line primitives.

the neighbor with a certain probability. These methods provide the search direction instead of it being just a random walk in the solution space. The local search algorithm moves iteratively from solution to solution. There are different termination criteria, for example a limit on the total number of iterations or a limit on the number of iterations in which no progress is made.

In the rest of this chapter, we first discuss possible representations of a feature that can be used in a local search algorithm. Also, we describe an approach for artificially determining the quality of a feature. Next we describe the choices that have to be made to implement a local search algorithm. Finally, we give an introduction to evolutionary algorithms. An evolutionary algorithm is a specific type of local search algorithm, which has the advantage over most other implementations of local search algorithms that it produces generations of features at the same time. This eliminates the need to determine the quality of a feature each time when a new feature is created.

7.1 Solution representation


Before we can apply a local search algorithm, we have to choose a suitable representation of a feature (solution). The most suitable representation may depend on the variables that are found most influential for the cellular response during the stepwise regression analysis. For example, if the variables WN_x representing the energy present in parts of the signal with wavenumber about x turn out to be most influential, a representation of a feature by its Fourier transform might be most natural.

When a feature representation is chosen, we can define operations that slightly change the feature. For example, if we keep the restriction that features are composed of primitives, we can represent a feature by a list of primitives with information on their type, position and rotation. We can then define operations that can be applied

to a feature to obtain a slightly modified feature, such as:

- Remove one of the primitives present in the feature.
- Add a new primitive to the feature at a random position and with a random rotation.
- Rotate one of the primitives present in the feature by a random amount.
- Move one of the primitives present in the feature to a new position.

Another possibility is to discretize a feature and to represent it by a binary matrix. The matrix elements that are equal to 0 represent elements of the feature that are low and the matrix elements that are equal to 1 represent elements of the feature that are high. A method to slightly modify a feature could be to switch each of the matrix elements from 0 to 1 or vice versa with a certain low probability.

A third option is to represent a feature by its values for the variables found influential during the stepwise regression analysis. A feature can be slightly modified by choosing a subset of these variables and slightly changing their values. 

When choosing a feature representation and a method to obtain new features by slightly changing an existing feature, we have to keep in mind that we have to be able to construct a feature that has the obtained feature representation. In some cases this is straightforward, for example using the representation of a feature as a list of primitives. If we represent a feature by a list of values for a collection of variables x_1, \dots, x_k however, this might be more difficult. Some of the variables x_1, \dots, x_k can be interrelated, which makes it impossible to create a feature for each combination of values for x_1, \dots, x_k . For example, assume x_1 represents the number of circle primitives used in the feature and x_2 represents the fraction of the feature covered by primitives. A nonzero value for x_1 requires a nonzero value for x_2 as well. For certain combinations of values for x_1, \dots, x_k , it is possible that no feature exists with these variable values. Even if one exists, it may be impossible to create in practice, if no algorithm is available to construct a feature from a feature representation. In some cases there may be multiple features that satisfy a particular feature representation. As long as all these features provide similar values for the cellular response this is not a big problem.

7.2 Determining the quality of a solution

Local search algorithms need information on the quality of features in the neighborhood in order to decide which new feature to move to. The most obvious way to determine the quality of a feature is to directly measure the cellular response for the feature. In our research however, this is a very time-consuming process. An alternative is to use an artificial quality measure based on the model found using the stepwise regression analysis. The stepwise regression algorithm terminates at a point at which all variables in the regression equation are significant, and all variables not in the regression equation are insignificant when added to the final model given by the regression equation. At this point, the regression equation is given by Equation (7.1).

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k \quad (7.1)$$

When we want to determine the quality of a new feature, we compute the variable values for x_1, \dots, x_k and enter these values in Equation (7.1). This gives us a value

for Y , which is used as an artificial quality measure for the feature. Since this process is very inexpensive and takes little time compared with doing an actual experiment to determine the quality of the features, it is possible to determine the quality of lots of new features. If at a later stage an experiment testing several feature is performed, we can use the results from this experiment to refine the regression equation by applying the stepwise regression algorithm to the variable values of the features in all of the generations.

Note that it is still necessary to perform actual experiments from time to time instead of determining the quality of features solely by the regression equation. One reason for this is that the regression equation is only valid for the range of the variable values that is present in the data set on which the regression equation is based. For example suppose that x_1 represents the number of circle primitives present in a feature and the regression equation is based on a data set in which x_1 takes values between 0 and 16. Now suppose that β_1 has a value larger than 0 in the regression equation. Adding more circle primitives increases the value for Y in the regression equation and if many iterations of the local search algorithm are applied, features with a large amount of circle primitives may be found with large values for Y . It can be expected that these features perform worse in practice than predicted by the regression equation, since the regression equation is not valid for large values of x_1 . Adding the 100th circle primitive to a feature that is already covered by 99 other circle primitives is unlikely to increase its quality as much as adding the first circle primitive to a feature. If an actual experiment is performed on some features with large values for x_1 and it turns out that these features are of lower quality than predicted by the regression equation, it is likely that the new regression equation has a lower value for β_1 . This leads the local search algorithm to search in other directions for improvement of features. It may be necessary to use a regression equation in which higher order terms for the variables x_1, \dots, x_k appear, since if these variables only appear linearly in the regression equation and they have a positive coefficient β_i it is optimal to increase the variable value as much as possible. This is unlikely to be optimal in practice for some types of variables.

7.3 Implementation of the local search algorithm

For the exact implementation of the local search algorithm a number of choices have to be made. These choices include which feature is used as a starting point for the algorithm (using multiple starting points is also possible), which operations are used to modify the features, and how many iterations of the local search algorithm are performed. In addition it may be desirable to allow changes that degrade the quality of a feature in some cases, to be able to escape from local optima. It is hard to determine in advance which kind of local search algorithm works well in our case. It is desirable to try multiple versions in order to select one that provides good results. Examples of local search algorithms include tabu search and simulated annealing. More information on local search algorithms can be found in the extensive available literature on the subject.

7.4 Evolutionary algorithms

Evolutionary algorithms are a certain type of local search algorithm. The advantage of evolutionary algorithms over other types of local search algorithms is that they work with generations of individuals (in our case these are the features), which are

all created and tested at the same time. This fits well with the process of testing the features by using a TopoChip. Lots of features can be tested at the same time, but it takes a relatively long time to get the results. Because of this it is desirable to be able to design each feature without the need for having the test results of the previously designed feature. This section describes the basics of evolutionary algorithms and comments on the suitability of evolutionary algorithms for improving the quality of the features. For a more elaborate introduction to evolutionary algorithms, see for example [8] and [9].

Evolutionary algorithms are based on the concept of the ‘survival of the fittest’, used by Charles Darwin in his theories on evolution. At each time point there is a collection of individuals. These individuals are vectors of decision variables in a search space of finite dimension. For example, each element of the vector can be a binary number, a real number, an integer, etc. Mixtures of different types of decision variables are possible as well. The collection of individuals forms the parent generation. The highest quality (also called fittest) individuals in the parent generation have the largest chance of being selected as parents, which is in line with the concept of the ‘survival of the fittest’. A child is created by choosing two (or more) parents and (randomly) recombining them to produce a new individual. This individual possesses some characteristics of both of the parent individuals. Sometimes (by chance) the child inherits different good properties of both of the parents, making it superior to its parents. Properties that are present in both parents are always present in the child as well. Since these properties are likely to be beneficial (because it is likely that fit parents are selected, which have mostly good properties), it is a good thing they are preserved in the child. The final step is the mutation step, in which some of the properties of the child created in the recombination step are modified randomly. This is done to provide a good property the chance to return in the next generation of individuals, when it is lost in a previous generation. This may happen for example, when the fittest individuals in the parent generation do not possess a certain good property by chance. Because these individuals are more likely to be selected as parents, the good property may disappear completely in the next generation.

As a small example of the working of an evolutionary algorithm, consider pancake recipes given by a triplet of nonnegative integers (M,E,S). These represent the number of liters of milk (M), number of eggs (E), and grams of salt (S) to add to a kilogram of flour. Pancakes are baked according to all of the recipes and judged by a professional cook, who gives a mark from 1 to 10 (10 being highest). Suppose we start with the recipes listed in Table 7.1. One way to select the first parent can be to select each recipe i with probability $p_i = \frac{m_i}{\sum_j m_j}$. In this equation m_j is the mark for recipe j . Similarly, the second parent can be selected. Note that better recipes have a larger chance to be selected. Suppose recipe (2,8,10) and recipe (4,4,20) are selected. The two recipes are recombined by taking the first k ($k = 0 \dots 3$) elements of the first recipe and the last $3 - k$ elements of the second recipe. Suppose $k = 2$. In this case the new recipe consists of the first two elements of the first recipe and the last element of the second recipe, which creates the new recipe (2,8,20). As a final step mutation is applied. Each element of the recipe is mutated by -1 with probability 0.1 (unless the element is 0), by +1 with probability 0.1, and by 0 with probability 0.8. Suppose this mutation step creates the recipe (2,8,19). This recipe is part of the next generation of recipes. In a similar way, multiple new recipes can be created. When a new generation of recipes is created, they can be rated by the cook, and the algorithm repeats with a new selection step.

Recipe	Mark
(2,8,10)	8.5
(4,4,20)	6.0
(1,3,10)	7.5
(5,2,30)	3.5

Table 7.1: The parent generation of pancake recipes and their quality.

There are lots of variants of evolutionary algorithms, but most of them follow the steps selection, recombination, and mutation. In the following a short overview of the different steps is given:

- *Selection.* Selection is the only step of the evolutionary algorithm that leads the search in the direction of promising regions of the search space. While in the recombination and mutation steps diversity is promoted, the selection step selects the fittest individuals as parents for the next generation. This can be done by selecting fitter individuals as parents with larger probability, like in the example given before. Another approach is to generate more children than there are parents and to keep only the fittest ones as possible parents for the next generation. In some cases the fittest individuals are only selected from the generation of children, but in other cases they can be selected from the parent generation, too. The last version makes sure that the fittest individuals are always preserved in the next generation. In practice however, depending on the type of the problem, both of the two versions may perform best.
- *Recombination.* Recombination is the process of constructing a new individual from a collection of parent individuals. In some versions of evolutionary algorithms this step is omitted and one of the parent individuals is used directly in the mutation step. There are two main types of recombination. Intermediate combination sets the value of each element of the new individual's vector equal to the average of the values of the same element in all of the parent individuals' vectors (in case of integers or binary numbers, rounding is necessary). Dominant recombination sets the value of each element of the new individual's vector equal to the value of the same element in the vector of one randomly selected parent (for each element a new parent is selected at random). As mentioned before, the main goal of the recombination step is to extract similarities from the parents, because these similarities usually represent beneficial properties.
- *Mutation.* Mutation is the process of applying (small) changes to an individual. This is done to promote diversity. There are lots of different variants of mutation and the best implementation depends on the specific problem considered. Most successful implementations of mutation obey the principles of reachability, unbiasedness, and scalability (see [9]). Reachability means that given a parent, each other individual can be reached within a finite number of mutation steps. Unbiasedness requires the mutation operator to be unbiased, that is it should base its mutations on the search space information of the parental population only. This requirement is based on the idea that selection should provide the search direction and mutations should provide maximum diversity. Scalability requires the mutation strength (average length of a mutation step) to be tunable. If the steps are too small, it takes many iterations to get close to the optimum. If the steps are too large, it is hard to get very close to the optimum, because

the algorithm 'skips over' the optimum. Preferably, the algorithm should take large steps when it is still far away from the optimum and small steps when it is close to the optimum. To ensure an implementation of an evolutionary algorithm satisfies the requirements mentioned above, advanced implementations of evolutionary algorithms typically work with individuals which besides decision variables and fitness information contain strategy parameters as well. These parameters determine the distribution of the mutations and evolve from generation to generation. A typical choice for real-valued search spaces is to use Gaussian mutations with center zero and standard deviation depending on the strategy parameters.

It is difficult to estimate how well evolutionary algorithms perform for our problem. The number of iterations of an evolutionary algorithm before a good solution is found, depends strongly on the problem type and the exact implementation of the algorithm. For an overview of the performance of several different evolutionary algorithms on a set of test problems, see [10]. From this overview, it appears that a couple of tens of generations of several thousands of features may be just enough to get close to an optimal solution, provided a suitable implementation of an evolutionary algorithm is chosen and the difficulty of our problem is comparable to the test problems. For most problems it is preferable to use a larger number of generations containing less features per generation, in order to find a solution of high quality in a limited number of function evaluations (measurement of cellular response in our case). In our case however, the most limiting factor is not the total number of measurements of cellular response we can make (one TopoChip can contain thousands of features), but the number of generations of features we can design. In terms of the number of generations needed to find a solution of high quality, it does help to increase the size of the generations.

The following reasons make evolutionary algorithms suitable for solving our problem:

- A large generation of features can be designed and for all these features the cellular response can be determined at the same time, without the need to use an artificial quality measure. This is an advantage over methods that need fitness information of an individual feature, before the next feature can be designed. Since in our case determining the quality of features takes a relatively long time, designing features in generations is preferable.
- Evolutionary algorithms do not make assumptions on the fitness landscape (how the fitness of an individual depends on its variable values). They work well in lots of different domains. Since we do not make any assumptions on the fitness landscape in our case, evolutionary algorithms seem suitable.

Evolutionary algorithms typically use not more than several tens of decision variables. This limits the possible solution representations that can be used in combination with evolutionary algorithms. For example, representing a feature by its 100×100 discretization produces 10000 decision variables. This is probably too high a number for an evolutionary algorithm to find a high quality solution in a limited number of generations.

Chapter 8

Conclusions

The production process that we use to create the TopoChip proves to be unable to create a TopoChip containing walls and features of the intended height and shape. This fact prevents us from measuring the cellular response for each of the features.

We use an artificial variable as the cellular response in the stepwise regression analysis. The stepwise regression algorithm succeeds in identifying the characteristics of the features that have an influence on the value of the cellular response. This is also the case if we first disturb the cellular response by adding a noise term. The more noise is added, the less variables are included in the regression model. A model that is based on a particular part of the data set is reasonably suitable for predicting the cellular response for another part of the data set. This suggests that the regression model does not contain many elements that are specific for the data set it is based on, and is applicable for all features that are created similarly.

We propose using local search algorithms as a method to create features to include in the next TopoChip. Local search algorithms seem suitable for this problem, since there are multiple possibilities for defining a solution representation and it is possible to define neighborhood solutions for each solution. The biggest problem is that determining the quality of a solution takes much time. A possible solution for this problem is to use evolutionary algorithms, which typically work with generations of solutions, for which the quality is determined simultaneously. Another possibility is to use an artificial quality measure. Which local search implementation is most suitable strongly depends on the specific problem considered. As long as no cellular response is available, it is difficult to decide which implementation works well.

Chapter 9

Recommendations

At the moment the number one priority is to have the cellular response available for each of the features. As long as no cellular response is available there are several uncertainties. First of all, it might be that the features have no influence on the cellular response. For example, it could be that they need to have a size of a different order of magnitude to have any influence. Also, without the cellular response we cannot determine which characteristics of a feature determine the value of the cellular response most. Depending on which characteristics are important, a different representation of a feature can be most suitable and a different implementation of a local search algorithm can work best.

When the cellular response is available, additional characteristics of the features can be formulated and used as independent variables in the regression analysis to see if these new variables are more influential in determining the cellular response. For example, new independent variables can be created by transforming other independent variables or by computing the product of two independent variables. It may be useful to have a person look at the highest quality features, to see if this person can identify any shared characteristics of these features that are not yet included as independent variables.

In advance it is usually difficult to predict which local search strategy will perform best for a certain problem. It might be useful to try multiple different search strategies simultaneously. This can be done for example by assigning each local search implementation a part of the TopoChip which it can use to try new features.

Chapter 10

Bibliography

- [1] Christopher J. Centeno, Dan Busse, John Kisiday, Cristin Keohan, Michael Freeman, and David Karli, *Increased knee cartilage volume in degenerative joint disease using percutaneously implanted, autologous mesenchymal stem cells*, Pain Physician **11:3**:343-353, 2008
- [2] K. Hatano, H. Inoue, T. Kojo, T. Matsunaga, T. Tsujisawa, C. Uchiyama, and Y. Uchida, *Effect of surface roughness on proliferation and alkaline phosphatase expression of rat calvarial cells cultured on polystyrene*, Bone **25**:439-445, 1999
- [3] Matthew J. Dalby, Nikolaj Gadegaard, Rahul Tare, Abhay Andar, Mathis O. Riehle, Pawel Herzyk, Chris D. W. Wilkinson, and Richard O. C. Oreffo, *The control of human mesenchymal cell differentiation using nanoscale symmetry and disorder*, Nature materials **6**:997-1003, 2007
- [4] Hayden Huang, Roger D. Kamm, and Richard T. Lee, *Cell mechanics and mechanotransduction: pathways, probes, and physiology*, American Journal of Physiology - Cell Physiology **287**:C1-C11, 2004
- [5] Rowena McBeath, Dana M. Pirone, Celeste M. Nelson, Kiran Bhadriraju, and Christopher S. Chen, *Cell shape, cytoskeletal tension, and RhoA regulate stem cell lineage commitment*, Developmental Cell **6**:483-495, 2004
- [6] John A. Rice, *Mathematical statistics and data analysis*, Belmont, Thomson: 3rd ed. 2007
- [7] Ruye Wang, *Two-dimensional Fourier transform*, http://fourier.eng.hmc.edu/e101/lectures/Image_Processing/node6.html, 2007
- [8] David E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, Reading, Massachusetts, Addison-Wesley 1989
- [9] Hans-Georg Beyer and Hans-Paul Schwefel, *Evolution strategies, a comprehensive introduction*, Natural computing **1**:3-52, 2002
- [10] Stefan Kern, Sibylle D. Müller, Nikolaus Hansen, Dirk Büche, Jiri Ocenasek, and Petros Koumoutsakos, *Learning probability distributions in continuous evolutionary algorithms - a comparative review*, Natural computing **3**:77-112, 2004

Appendix A

Parameter values and ranges

Table A.1 provides an overview of all parameter values and ranges used to construct the features in the first generation. Some parameters have different ranges for each feature size. These parameters are listed separately for each feature size, with the length of one side of the feature given in brackets. See Section 4.3 for more information on the parameters.

Parameter	Value or range
Feature side length	10 μm , 20 μm , or 28 μm
Number of primitives used (10 μm)	3 - 5
Number of primitives used (20 μm)	3 - 12
Number of primitives used (28 μm)	3 - 16
Diameter of a circle primitive (10 μm)	3.0 μm - 4.0 μm
Diameter of a circle primitive (20 μm)	3.0 μm - 10.0 μm
Diameter of a circle primitive (28 μm)	3.0 μm - 10.0 μm
Shortest side length of a triangle primitive (10 μm)	3.0 μm - 4.0 μm
Shortest side length of a triangle primitive (20 μm)	3.0 μm - 8.0 μm
Shortest side length of a triangle primitive (28 μm)	3.0 μm - 10.0 μm
Top angle of a triangle primitive	36°
Length of a line primitive (10 μm)	3.0 μm - 8.0 μm
Length of a line primitive (20 μm)	3.0 μm - 16.0 μm
Length of a line primitive (28 μm)	3.0 μm - 23.0 μm
Thickness of a line primitive	3 μm
Standard deviation for the rotation of a primitive	0.0° - 180.0°

Table A.1: The range or value for each of the parameters used to construct the features in the first generation

Appendix B

Independent variables used in the regression analysis

Several independent variables are used in the regression analysis. See Table B.1 for a short overview. A more detailed description of the variables is given in Section 5.2.2.

Variable	Description
<i>NUM</i>	The feature number
<i>ROW</i>	The row number of the feature on the TopoChip
<i>COL</i>	The column number of the feature on the TopoChip
<i>DC</i>	The density of circle primitives
<i>DT</i>	The density of triangle primitives
<i>DL</i>	The density of line primitives
<i>CA</i>	The area of circle primitives scaled with their density
<i>TA</i>	The area of triangle primitives scaled with their density
<i>LA</i>	The area of line primitives scaled with their density
<i>ROT</i>	The scaled standard deviation of the rotation of primitives
<i>CCD</i>	Number of color changes of the feature over the diagonal
<i>WN</i> _{0.1}	The fraction of energy in the signal with wavenumber ca. 0.1
<i>WN</i> _{0.2}	The fraction of energy in the signal with wavenumber ca. 0.2
<i>WN</i> _{0.3}	The fraction of energy in the signal with wavenumber ca. 0.3
<i>WN</i> _{0.4}	The fraction of energy in the signal with wavenumber ca. 0.4
<i>WN</i> _{0.5}	The fraction of energy in the signal with wavenumber ca. 0.5
<i>WN</i> _{0.7}	The fraction of energy in the signal with wavenumber ca. 0.7
<i>WN</i> ₁	The fraction of energy in the signal with wavenumber ca. 1
<i>WN</i> _{1.5}	The fraction of energy in the signal with wavenumber ca. 1.5
<i>WN</i> ₂	The fraction of energy in the signal with wavenumber ca. 2
<i>WN</i> ₃	The fraction of energy in the signal with wavenumber ca. 3
<i>WN</i> ₄	The fraction of energy in the signal with wavenumber ca. 4

Table B.1: The independent variables used in the regression analysis