Shining a Light on Material Appearance

Mapping NDFs to Heightfields

Casper Struijk[1]

Supervisor(s): Ricardo Marroquim[1]**, Yang Chen**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Casper Struijk
Final project course: CSE3000 Research Project
Thesis committee: Ricardo Marroquim, Yang Chen, Daniël Pelsmaeker

Abstract

This research proposes a new algorithm for mapping Normal Distribution Functions to Heightfields in order to answer its research question: "Given an NDF, how can we generate a corresponding Heightfield using simple optimization algorithms?". This research is important, as it helps us to gain a better understanding of how limited statistics-based representations of 3D surfaces are. To this end, we have produced an algorithm using the Simulated Annealing optimization technique, a technique that randomly explores possible solutions of a problem until it finds the optimal solution. The algorithm begins with a flat Heightfield (a 2D representation of a surface that shows the relative altitude of a discrete plane of points), iteratively changes points on the Heightfield and compares its measured NDF (Normal Distribution Function, a function to denote the area distribution along a given direction in a Heightfield) to the target NDF that we want to map. Once the target NDF is reached, or once the pre-determined number of iterations has been reached, the algorithm concludes and the NDF-to-Heightfield mapping has been completed. Three different variations are tried, one naïve implementation which changes points on the Heightfield completely at random, another where the angle of the normal vector of a random surrounding facet of a chosen point is used as guidance for randomization, and finally one where this angle is guided using the relative position of the chosen point to the centre of the Heightfield. The conclusion the proposed algorithm provides is that, while possible, the process of mapping NDFs to Heightfields is a costly and complex operation, and leaves a lot of room for ambiguity. While the research cannot provide a case of an exact match of a target NDF and measured NDF of a Heightfield created through the algorithm, we do show it is without a doubt possible given time.

1 Introduction

Over the course of decades, computer graphics technology has had significant advancements. In this day and age, graphics are no longer limited to 2D, and no longer do 3D graphics look rough and polygonal. With the introduction of techniques such as photo-realistic rendering, video games and computer simulations now look more realistic than ever. But this technology is anything but easy to understand when you look past its surface. 3D graphic rendering has a lot of complex science hiding within, from photodynamics to complex mathematical models. As this technology improves and new rendering techniques are discovered, this science becomes even more complex, as over time we gain a better understanding of how to simulate reality inside a computer.

This research will focus on two essential aspects of computer graphics. One of these is Heightfields, also referred to as Heightmaps. Heightfields are 2D representations of surfaces of 3D objects or in some cases small portions thereof. Commonly Heightfields are represented as images, otherwise known as maps, that represent the differing heights of points in the map by making use of lighter or darker colours for higher or lower points respectively. Optionally, these 2D Heightfields can be mapped into 3D meshes to further help visualise their representation and make it more understandable. Below in Figure 1 is an example of a 2D Heightfield for reference.

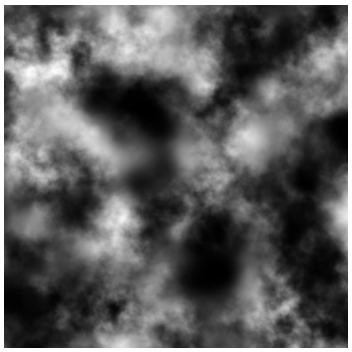


Figure 1: A 2D Heightfield. In this grayscale map, lighter colours represent higher altitudes of the points in that area, and darker colours represent lower altitudes.

The other is NDFs or normal distribution functions [5], not to be confused with the term "normal distribution" from statistics theory. NDFs are functions that denote the distribution of the angles of the normal vectors of the facets that make up the surface. The focus of this research will especially be on the latter, as we aim to mitigate the flawed nature of NDF representations of surfaces. Namely, because NDFs are a statistical representation, the actual appearance of the surface is often not fully apparent. That is to say, while for example, one Heightfield will always translate into one and the same NDF, one NDF may be representative of any number of Heightfields, some of which may very well vary wildly from one another.

In order to mitigate the lacking nature of NDF representation, the aim of this research is to introduce a new algorithm for mapping NDFs to Heightfields, as Heightfields can help us gain a better picture of the surface the NDF is meant to represent. As such, the main research question for this research is as follows: "Given an NDF, how can we generate a corresponding height field using simple optimization algorithms?" This question will be answered with the help of two smaller sub-questions, namely: "What optimizations can be applied to existing algorithms?" and "How should ambiguity be handled in the algorithm?"

The reason that this research is important, is that this algorithm can provide us with keen insight into the limitations of statistical graphics models like NDFs, such as the abovementioned issue of NDFs mapping to potentially multiple Heightfields. The algorithm that this research will propose may also serve as a new technique for rendering surfaces, which could prove to be useful to the advancement of our

understanding of 3D graphics.

Presumably due to this being a somewhat niche research topic, not a lot of relevant literature can be found on the relevant topics of this research. However, some research has already been done on this topic. An example of such is the paper written by M. Ribardière et al. [6], where they describe the impact NDFs have on the appearance of objects and how they can be used to simulate a variety of effects on surfaces. They also mention, however, that their model for mapping NDFs to BRDFs (Bi-directional Reflectance Distribution Functions) [4] is incomplete, as it cannot for example account for energy loss in particularly rough surfaces. Another valuable by T. Weyrich et al. [8] shows their implementation of a statistic graphics model mapping to Heightfields. However, in their research, they do not use NDFs as their model but use BRDFs instead.

This report is structured as follows. In Section 2, the research methodology is discussed, including the different optimization techniques that were considered in the making of the algorithm. Section 3 details all the results obtained throughout the research, both intermediate results as well as final results. After this, Section 4 features discussion of the results, as well as discussion of various other topics such as the limitations of the research and the ethical decisions made during the research process. A list of references to other material will be provided at the end.

2 Methodology

In order to come up with a new and efficient algorithm for NDF to Heightfield mapping, first a base algorithm is needed to make sure the mapping task can be accomplished. An algorithm for mapping Heightfields to NDFs has been provided by the supervisor for this course, Y. Chen. Next, this provided algorithm must be used in a bigger algorithm that will synthesize a Heightfield to match a given NDF. This algorithm takes an NDF that the user provides, and through an optimization process creates a Heightfield whose NDF matches the one provided by the user. The algorithm that is left at the end, taking an NDF and giving back a Heightfield, will help answer the main question of this research, by showing that it is indeed possible to map NDFs into Heightfields. The sub-questions will be answered in the process, and thoroughly discussed in the later sections of this paper.

2.1 Simulated Annealing

The optimization technique that is considered is called Simulated Annealing [1]. Annealing is a term that hails from metallurgy and refers to the gradual cooling of a hot metal in order to manipulate its shape. Simulated Annealing (after this referred to as SA) in the context of computer science refers to an optimization technique that can be used to solve complex problems, particularly combinatorial problems. An SA algorithm works by exploring the search space of a problem, in other words, it explores possible solutions to the problem. Generally, during every iteration, the algorithm will attempt to choose a better solution than its current known best solution. However, if an explored solution is worse than the current best solution, there is a probability that the algorithm will accept it. This probability gradually becomes smaller as time goes on, eventually converging on a probability of zero. The nature of this probability simulates the slow cooling technique of annealing in metallurgy.

The reason SA is chosen as the optimization method for the algorithm is its ability to accept “lesser” solutions in pursuit of an optimal solution. Many optimization algorithms tend to get stuck on local optima, in other words, solutions that may be the maximum within a certain part of the search space but not the overall optimal solution. SA avoids this by being able to choose solutions that are worse than its current optimum, thereby finding the global optimum in most cases. As NDF mapping is a complex optimization problem, SA is deemed the most fitting technique to improve the mapping algorithm. In making this SA implementation, inspiration is taken from E. Howell’s implementation of SA for the Travelling Salesman problem [3]. However, much of his implementation has been changed, due to a difference in problem application. Only the general structure of the code logic remains.

2.2 Basic Implementation

Having selected SA as our optimization strategy to implement, we want to test different iterations of the algorithm, which are variations of a basic algorithm. The idea of the basic algorithm is to work in simple steps. The algorithm starts with a flat Heightfield, and a target NDF to reach. It then measures the NDF of the current Heightfield and compares it to the target NDF. If the current NDF matches the target, then the surface is correct, and the Heightfield has been completed, thus terminating the algorithm. If the NDFs do not match, however, the points of the Heightfield are altered by examining its neighbouring states. A ”neighbouring state” of a given Heightfield is considered a Heightfield with a difference of one and only one point. This one point is randomized with a new value during the point randomization step. For the newly generated Heightfield, the NDF is once again measured and checked against the target. If the target and measured NDF match, the algorithm terminates, otherwise the iteration continues.

Generally, we want the algorithm to accept the NDF that is closest to the target. However, because of how SA works, there is a probability the algorithm will accept the worse solution between the measured NDF of the current step and of the previous step. This probability is determined by a ”temperature” variable, meant to simulate its counterpart in metallurgy, that slowly decreases over the iterations of the algorithm. The lower the temperature, the lower the probability of the algorithm accepting a worse solution to iterate over next. Naturally, the temperature, and thereby the associated probability, will reduce to zero over time, thus allowing the algorithm to gradually converge on a globally optimal Heightfield.

At every step, to determine how close a measured NDF is to the target NDF, the mean squared error is calculated between the two graphs. The lower the mean squared error, the closer the graphs are to one another. Using this value, it can be determined which of two measured NDFs is closer to the target, namely, the measured NDF with a lower mean squared

error is the better solution.

The algorithm has two conditions for terminating. If during any step the measured NDF matches the target perfectly, in other words, if the mean squared error is 0, the algorithm stops and the Heightfield created during that iteration is returned as the result of the algorithm. Otherwise, the algorithm will terminate after a set number of iterations. This number is a global variable in the program and can easily be customised to be bigger or smaller.

2.3 Theta implementation

The next iteration of the algorithm builds off the basic implementation and adds the use of theta angles. These are the angles of the normal vectors of a Heightfield's facets with the flat XY plane. A high theta angle means a facet is very flat, and a low theta angle means a facet is very steep. These angles are used in the point randomization step of the algorithm.

When a new Heightfield is created, the differences between its values for every possible theta angle are calculated, and stored in an array we refer to as the "difference array". During the point randomization, a random facet surrounding the chosen point is selected, and the theta angle is calculated. From the difference array, the theta with the biggest difference is selected and is compared to the theta angle of the normal vector. If the vector's angle is lower than the theta in the difference array, the point is lowered by a random amount to make the normal vector's angle more steep. Conversely, if the vector's angle is higher, the point is lowered by a random amount to make the normal vector's angle flatter. By changing the point like this, the normal vectors of all the other surrounding facets change as well.

After the point is changed, the algorithm continues as before, by calculating the mean squared error of the newly changed Heightfield's NDF and the target is calculated, and the algorithm follows the logic as described in the basic implementation.

2.4 Phi implementation

The final version of the algorithm builds on the theta implementation by making use of another type of angle in the Heightfield, the phi angle. By looking at the position of the point chosen during the randomization step on the XY plane relative to the centre point of the Heightfield, the phi angle is determined based on their relative position, as shown in Figure 2.

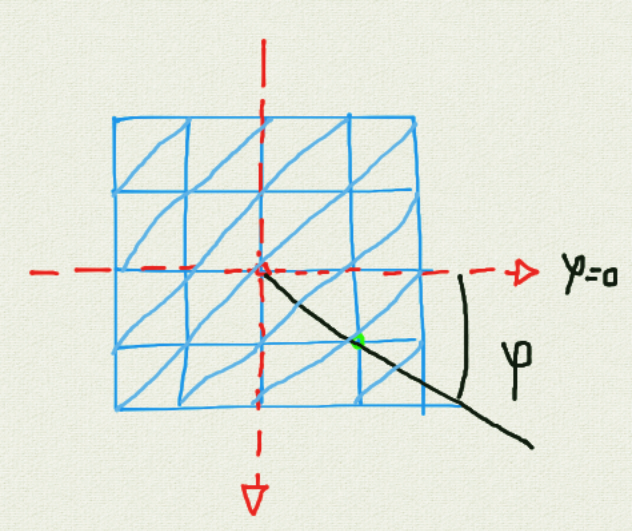


Figure 2: A visual guide to the calculation of the phi angle. The green point represents the randomly chosen point, and the red point in the middle is the centre of the Heightfield.

The main change this version of the algorithm makes is during the randomization step, once the random point has been chosen, a random surrounding facet has been chosen, and the theta angle of the normal vector has been calculated. Rather than determining the theta with the maximum difference in the entire difference array, instead the theta with the maximum difference that has the same phi angle as the chosen point is determined. This is done to make the randomization process smarter and to try to match the target NDF faster. Beyond this change, the algorithm remains the same as the theta implementation.

3 Results

This section of the paper presents all the results of the research, both intermediate results and final results. Where applicable, graphics will be provided.

3.1 Process

Before implementing the SA algorithm, some initial testing is done on the algorithm to map Heightfields to NDFs that is provided by Y. Chen. This is an integral component of the SA algorithm, as it would need to calculate the NDFs of a Heightfield at every iteration. Some simple Heightfields are picked as candidates, and all yield clearly defined NDFs. Figure 3 is an example of one of the NDFs that is produced from this initial testing progress.

Having figured out how the Heightfield-to-NDF conversion works, one Heightfield is chosen to take an NDF from. The Heightfield in question is a small 8x8 Heightfield, specifically chosen for its small size to reduce the amount of computation needed during tests and to make the tests faster as a result. The NDF of this Heightfield is the target NDF for all remaining tests described in this paper.

First, the basic implementation is tested. Several runs of the algorithm are done, as the outcome is highly random due

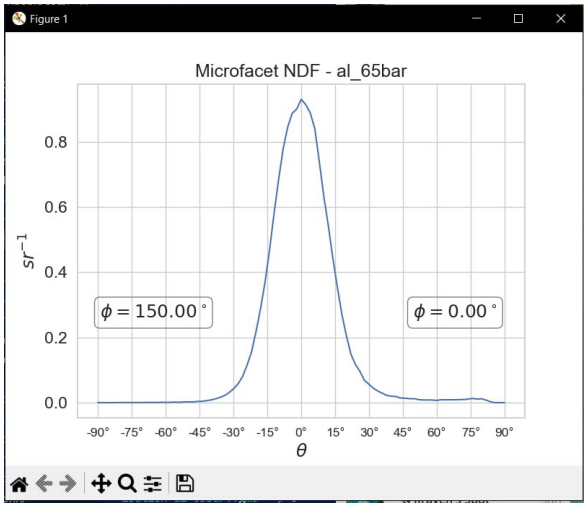
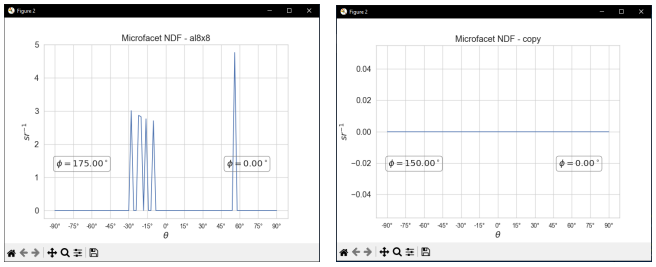


Figure 3: An NDF measured from a Heightfield. On the x-axis are the angles between the normal vectors and the level plane, and on the y-axis are the relative frequencies of the angles.

to the nature of the optimization technique. For all of these runs, the maximum cap in iterations was set to 1000, the temperature variable set to start at 1000, and the gamma factor set to 0.99. This means that for each of the 1000 iterations, the temperature is multiplied by 0.99 to slowly lower it. All of the resulting NDFs are quite distant from the target, as the actual optimization that the algorithm performs is quite random. One of the resulting NDFs is shown in Figure 4.

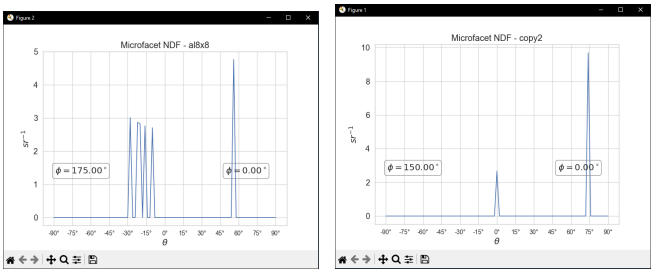


(a) Target NDF. (b) Measured NDF.

Figure 4: A resulting NDF of the basic implementation. On the left is the target NDF, and on the right is the measured NDF. The x-axis denotes the theta angles of the normal vectors, and the y-axis denotes the relative frequencies of these angles.

As can be clearly seen, the measured NDF is flat, likely due to this version of the algorithm not properly altering the Heightfield and thereby causing the Heightfield-to-NDF algorithm to give junk data as a result.

After this test, the next tests performed are the tests on the theta version of the algorithm. Once again the same target NDF is used, but now the theta angles of the Heightfield are taken into account during the point randomization step of the algorithm. The number of iterations, the temperature, and the gamma factor are still kept the same from the previous round of testing. This version is more computationally expensive, as such the performance of this algorithm turns out to be significantly slower than the basic version. The results of these tests are shown below in Figure 5.

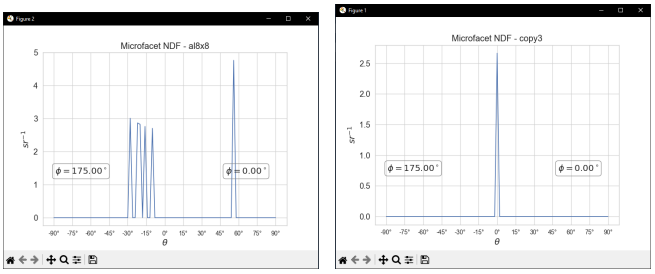


(a) Target NDF. (b) Measured NDF.

Figure 5: A resulting NDF of the theta version. On the left is the target NDF, and on the right is the measured NDF. The x-axis denotes the theta angles of the normal vectors, and the y-axis denotes the relative frequencies of these angles.

Comparing this measured NDF to the one shown in Figure 4b shows that this algorithm is a significant improvement. However, these NDFs clearly do not match yet, so this implementation is still incomplete.

For the third round of tests, the phi version of the algorithm is examined. Most of the variables are kept the same, but this time the gamma factor is upped to 0.999. In the past two rounds of tests, the mean squared error score of the measured NDF converged rather quickly, which theoretically could withhold the algorithm from finding a better optimal solution. As such, by upping the gamma factor, this convergence is much slower. Once again, a few tests are run to mitigate the randomness, and a sample solution is selected, which is listed in Figure 6.



(a) Target NDF. (b) Measured NDF.

Figure 6: A resulting NDF of the phi version. On the left is the target NDF, and on the right is the measured NDF. The x-axis denotes the theta angles of the normal vectors, and the y-axis denotes the relative frequencies of these angles.

This implementation's results confusingly seem to drift further away from the theta implementation again, as the NDF looks less similar to the target than the NDF shown in Figure 5b, despite this iteration of the algorithm more cleverly making use of the Heightfield's data to correct its exploration of the search space. To obtain more information on what is

causing this difference, the Heightfield that is produced by the algorithm for the above-shown NDF is examined. This Heightfield is shown in Figure 7.

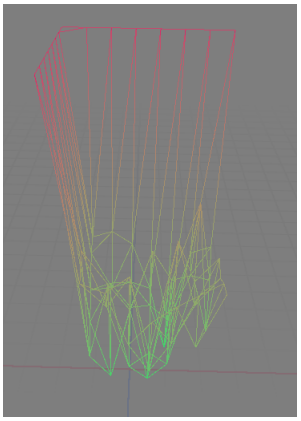
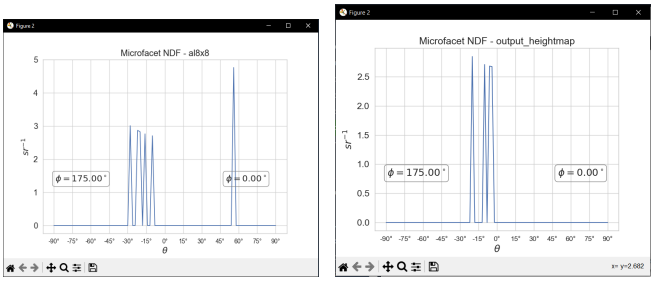


Figure 7: The output Heightfield, rasterised and mapped in 3D for better visibility. In this figure, green hues represent low altitude points, and red hues represent high altitude points.

Something that immediately jumps out while looking at this Heightfield is the oddly straight edges on the left and at the top. This is likely due to the algorithm somehow not reaching these points with its randomization, thereby also preventing the algorithm from reaching a matching NDF. With this in mind, a change is made to how the point randomization is done by altering the range of points that can be randomized. This leads to the test described in Section 3.2.

3.2 Final results

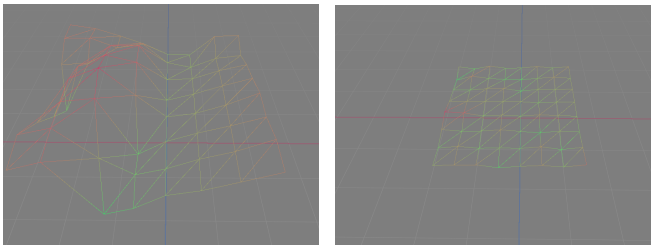
To do the final round of tests, some tweaks are made to the phi implementation. The maximum cap on iterations is increased to 5000 to allow the algorithm to further optimize the Heightfield it creates. The temperature is raised as well to 1500, to compensate for the higher iteration count. The gamma factor is kept at 0.999 still. The last, and possibly most important change, is the number by which a point can be changed per step from the interval [0, 0.5] to the interval [0, 0.1]. This allows the algorithm more precise steps and reduces the risk of overcompensating a point and accidentally increasing the mean squared error score. After again running this algorithm a small number of times, the best solution is picked and is shown here in Figure 8.



(a) Target NDF. (b) Measured NDF.

Figure 8: A resulting NDF of the final algorithm version. On the left is the target NDF, and on the right is the measured NDF. The x-axis denotes the theta angles of the normal vectors, and the y-axis denotes the relative frequencies of these angles.

Once again, for this version, the rasterized Heightfield is also examined and compared to the Heightfield that the target NDF is based on. These are shown below in Figure 9.



(a) Target Heightfield. (b) Produced Heightfield.

Figure 9: The original Heightfield and the output Heightfield, rasterised and mapped in 3D for better visibility. In this figure, green hues represent low altitude points, and red hues represent high altitude points.

As can be seen, these Heightfields don't differ a lot, and the NDFs are quite similar as well, though they aren't quite the same either. While due to time constraints, it is not possible to show in this paper, theoretically if this algorithm would run for much longer, perhaps say 300000 iterations, then the mean squared error between the target and measured NDF would eventually converge on 0. For the above example, the mean squared error started at 1454, and ended on its last iteration at 1.55, with its lowest mean squared error at 0.65. This means that while slow and computationally expensive, the algorithm does in fact work.

4 Discussion and conclusions

This section discusses the results of the research and presents the conclusion to the research question. First, the limits of the research will be discussed, both in terms of time and material. In discussing the results, possible improvements to the current research and future research ideas will be described, as well as how the results align with expectations. Furthermore, it will be discussed how this research improved on the existing material. There will also be discussion on the ethics

of the research performed. Finally, a summary of the research will be given and the research questions will be answered.

4.1 Limitations and improvements

A major limitation of this research is the complexity of implementing optimizations into NDF mapping algorithms. A lot of existing NDF mapping algorithms, such as the algorithm proposed by M. Ribardière [6], are highly complex, and adding an algorithmic optimization to this is a very complicated process. Even further, once the algorithm has been implemented, it is very difficult to debug or properly test its functionality, as you have to rely solely on the data you receive from running different versions of your algorithm. As such, a lot of the implementation for this research proves to be quite time-consuming, thus leading to only being able to implement an SA optimization in this algorithm, rather than being able to test multiple different techniques. This way, the results of the SA algorithm end up being much greater simply because it has more time invested in it.

Another limitation on specifically an algorithm optimizing using SA is the nature of its randomness in conjunction with the size of the search space of this particular problem. The amount of different states that even a low-resolution Heightfield can have is nigh uncountable, due to the height values of the points on the Heightfield not being discrete integers, but rather real numbers. Theoretically, if this SA algorithm was allowed to run not for 1000 iterations, but for example 100000 iterations, the probability of finding a true global optimum would be much greater. However, this would be incredibly costly in terms of time. On an 8x8 resolution Heightfield, 1000 iteration tests could sometimes take up to 2 minutes to complete. This number would increase exponentially the higher the Heightfield's resolution due to more calculations needing to be done at every single step. Therefore, these longer tests were not performed in this research due to time constraints.

Lastly, one limitation pertaining to the conversion from an NDF to a Heightfield, as briefly mentioned in Section 1, is its ambiguous nature. One NDF can map to theoretically infinite Heightfields. As the points of a Heightfield do not have discrete integer values for their heights, but can have any real number as a value, there are an infinite number of Heightfields that can map to one NDF. As such, a proper NDF-to-Heightfield mapping algorithm will still return random results using an optimization technique such as SA. And in a realistic context, where one would have their 3D surface before they had their NDF, it would be unlikely that the Heightfield produced by an NDF-to-Heightfield mapping algorithm would match the surface the user already has.

4.2 Future work

The SA implementation of this NDF-to-Heightfield mapping algorithm is, as it currently stands, quite slow at its calculations. As mentioned above, 100000+ iteration tests could take several hours with its current implementation even on low-resolution Heightfields. Therefore, a good improvement on this algorithm would be to improve its efficiency.

SA is of course not the only optimization strategy that is applicable to this NDF-to-Heightfield mapping problem. In the course of studying for this research, more optimizations were found that could be theoretically used in a mapping algorithm. A prime example of this is the Branch and Bound algorithm [2]. This algorithm is a spin on the Divide and Conquer algorithm paradigm [7]. Splitting a problem into different possible sub-problems and using its ability to discard subsets of problems if they are not likely enough to produce satisfactory results could well lead to a much faster algorithm, its supposed downside being that it does not have the resilience to getting stuck local optima that SA has. Despite the latter comment, I believe this is a worthwhile topic to research.

4.3 Expectations versus reality

After having chosen SA for optimizing NDF-to-Heightfield mapping, it was expected it would perform quite strongly in this task. The nature of NDF mapping, namely a complex combinatorial problem with potentially many local optima, makes SA a perfect candidate to optimize our algorithm, due to its ability to avoid getting stuck on local optima and search beyond them to find a global optimum. However, the results are not as positive as I hoped. My expectation that the algorithm would provide data was somewhat naïve, as I did not initially realise how computationally expensive this problem is.

Talking more about the inner workings of the algorithm, initially when SA was chosen as a candidate algorithm, a specific implementation idea was drafted for how the algorithm would function. In this draft, the initial Heightfield would be a random set of points, rather than a flat Heightfield, and the algorithm would then make random changes to the Heightfield and record the effect it has on the measured NDF, whether or not a change would cause it to be more or less accurate. However, while this was the initially expected implementation, the reality ended up quite a bit different due to some information provided by Y. Chen. The random starting Heightfield was scrapped in favour of the flat Heightfield, and the iterative step was adjusted to be more in line with SA paradigms.

4.4 Research contribution

This research has managed to provide in two distinct ways. For one, it has proposed a new algorithm for mapping NDFs to Heightfields that, while not flawless, with some improvement can prove to be very effective in its application. Beyond that, this research provides evidence for the claim that statistical models for 3D surface representation such as NDFs are unreliable when it comes to effectively representing surfaces.

By looking at the results in Section 3, you can see that the algorithm does not quite manage to create a Heightfield with a perfectly matched NDF as its target, given that this problem is incredibly complex and computationally intensive. Also taking into as well account that an NDF on its own does not tell us much about the actual look of our surface, given we do not know where the normal vectors it denotes lie on the surface, it is clear that this statistical model is lacking in its expressiveness and robustness.

4.5 Responsible research

In the course of this research, ethics was constantly considered. In all cases where code made by other people was used, proper credit was given, and an explanation was provided as to precisely what had been taken from their implementations. Likewise, any and all references to other literature over the course of writing this paper have been carefully compiled. The code for this research is available online, thus the research is both reproducible and extensible by anyone that wishes to continue or improve the project this research has begun.

4.6 Summary and conclusion

To answer the research questions, starting with the sub-questions: The answer to the sub-question "What optimizations can be applied to existing algorithms?" is, in short, a number of different optimization techniques. This research has only implemented Simulated Annealing and has posited Branch and Bound as another option, but there are many theoretical options beyond these two. For the second sub-question, "How should ambiguity be handled in the algorithm?", the answer is that ambiguity is an unfortunate but inevitable reality. Because NDFs and Heightfields do not have a 1-to-1 relationship, there is no guarantee that an NDF-to-Heightfield mapping algorithm will always produce the same result, thus leaving room for ambiguity.

To answer the main research question, "Given an NDF, how can we generate a corresponding height field using simple optimization algorithms?": Theoretically, yes. If you would let the above-described algorithm run for long enough, it should be able to produce a Heightfield that has an NDF perfectly matching its original target. However, while this is theoretically possible, this research has unfortunately not definitely proven it as fact.

To conclude, this research set out to propose a new algorithm for mapping NDFs to Heightfields, and to show that statistically-based 3D surface representation models such as NDFs are ambiguous and unreliable methods for representing 3D surfaces. The algorithm the research provides, while not wholly perfect, provides a working implementation to accomplish the task the research question set out to accomplish, and at the same time provides a stepping stone for further research to improve the algorithm's efficiency.

References

[1] E. H. L. Aarts and P. J. M. Laarhoven. Simulated annealing: An introduction - aarts - 1989 - statistica ... *Wiley Online Library*, 1989.

[2] J. Clausen. Branch and bound algorithms - principles and examples. *Janders EECG*, Mar 1999.

[3] E. Howell. How to solve travelling salesman problem with simulated annealing. *Medium*, Apr 2023.

[4] S. R. Marschner, S. H. Westin, E. P. F. Lafortune, and K. E. Torrance. Image-based bidirectional reflectance distribution function measurement. *Applied Optics*, Jun 2000.

[5] M. Olano and M. North. Normal distribution mapping. *CSEE UMBC*, 1997.

[6] M. Ribardière, B. Bringier, L. Simonot, and D. Meneveaux. Microfacet bsdfs generated from ndfs and explicit microgeometry. *ACM Digital Library*, Jun 2019.

[7] D. R. Smith. The design of divide and conquer algorithms. *Science Direct*, Mar 2003.

[8] T. Weyrich, P. Peers, W. Matusik, and S. Rusinkiewicz. Fabricating microgeometry for custom surface reflectance. *ACM Digital Library*, Jul 2009.