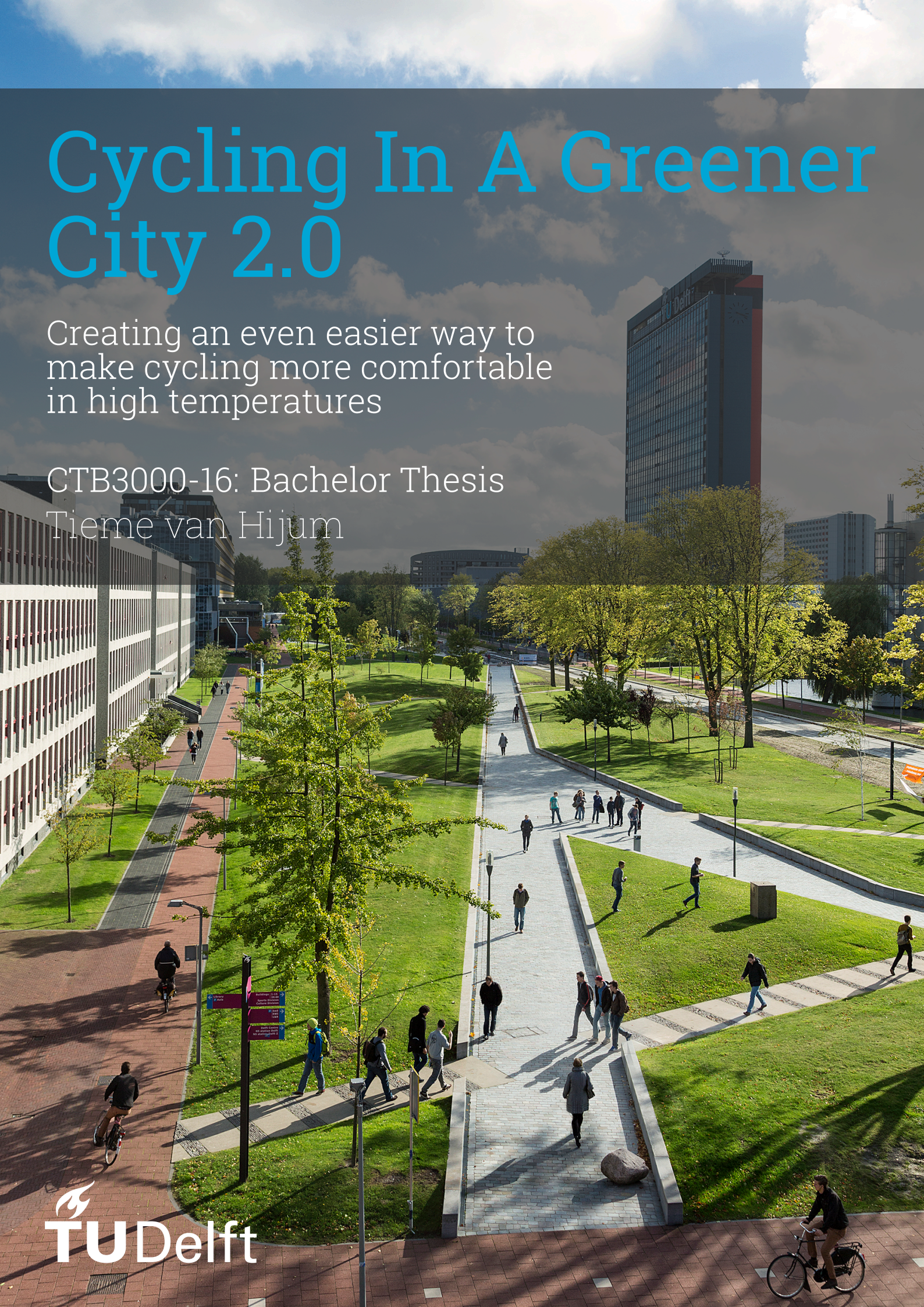
Cycling In A Greener City 2.0

Creating an even easier way to make cycling more comfortable in high temperatures

CTB3000-16: Bachelor Thesis

Tieme van Hijum



TU Delft

Cycling In A Greener City 2.0

Creating an even easier way to make cycling more comfortable in high temperatures

by

Tieme van Hijum

to complete the Bachelor Thesis for Civil Engineering
at the Delft University of Technology,

Student number:	4923588	
Project duration:	September 4, 2023 – October 23, 2023	
Department:	Transport & Planning	
Supervisors:	Dr. Y. Yuan,	Supervisor & Examiner
	Dr. S. Sharif Azadeh,	Supervisor & Examiner
	Dr. Ir. K. Kavta,	Supervisor

Cover:	TU Delft Campus by TU Delft from https://filelist.tudelft.nl/TUDelft/Over_TU_Delft/Guest%20Centre/Vierkant%20Virtual%20Campus2.png
Style:	TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft

Preface

This report is meant to be read by people who are interested in making cycling more comfortable in urban environments. No prior knowledge is needed to understand this report. However, some knowledge on GIS and Python would be make it easier to reproduce and improve the results of this thesis.

The research is built upon a previous report by Steven Barendregt, who I would like to thank for the provided data. Further, I want to thank my supervisors Dr. Y. Yuan, Dr. S. Sharif Azadeh and Dr. Ir. K. Kavta for their guidance during the writing process of this report. I want to thank my peers Arend-Jan Timmermans, Antoon Poelmans and Daniël Rijnders for their weekly feedback.

Tieme van Hijum
Delft, October 2023

Abstract

In The Netherlands, high temperatures in summer make it uncomfortable for cyclists. The most comfortable places to cycle during those high temperatures are in the shadows from trees or buildings. Making sure there is enough shade along cycling paths in urban areas is essential to make sure people keep cycling in the city. Otherwise, more people could be taking less sustainable modes of transport, like the car.

To make sure enough shade is provided by trees, Steven Barendregt (2023) conducted research on how to calculate shadow coverage using GIS data. He found out how to use Python to calculate the percentage of cycling paths in Delft which are covered by shade from trees. For this, he used tree data and cycling path data provided by the municipality of Delft. He first calculated the shadow coverage from all original trees. After that, he newly placed more than 200 standard trees into the data, by hand, in QGIS. This manual placement is, firstly, a tedious task, as it involves some repetitive work. Secondly, the results were not realistic for how trees would normally be placed, as he did not focus on that part for his research. This report aims to improve this process by making it more automated and realistic. Also, anyone should be able to pick up where this report leaves off and easily achieve the same results. Therefore, the research question for this thesis is: 'What is the optimal way to automate and improve the process to provide more shade in high temperatures for cyclists in Delft?'

This report's results are a new way to calculate shadow coverage, for which the user only has to give some simple inputs. The system will then, automatically, calculate or place trees, based on the user's wishes. This is achieved by using Python notebooks and ArcGIS Pro models. The first model does the same calculations as Steven Barendregt (2023) for the original tree data. It makes sure a new user can input original tree data and make calculations, without having to do these themselves, and thus, having to figure out how these calculations work. Next, a Python notebook is used to calculate the shadow coverage. All the user has to do is make sure the first cell is filled in with the correct inputs. Then, the Python notebook will automatically do the necessary calculations and give the percentage of shadow coverage from the original trees. The next step is the new process for placing new tree data. Some preparations have to be done, using another small model, which is the only manual work that has to be done in ArcGIS Pro. After that, a third model is used to automatically place new trees. These new trees can be in four different sizes. After that, a separate Python notebook can be used to perform the same shadow coverage calculations as for the original trees, but now with the new trees.

This report achieved its goals to automate and improve the process. Using the models and Python notebooks created for this report, anyone that wishes to continue this research will be able to do this with the updated process. However, further research still needs to be done in this subject. For instance, the results are still not the most realistic. While the user can choose from four different tree sizes, these tree sizes are based on some assumptions made from limited information. In further research, making sure the data becomes more realistic could be a priority. Another thing suggested by Steven Barendregt (2023) was to research what percentage of shadow coverage is desirable. As the actual value was not useful for this report, just the process to get to that percentage, no further research was done for this.

Contents

1 Introduction

Cyclists in The Netherlands are faced with high temperatures on a more regular basis. According to De Kruijf et al. (2021), "...weather at low air temperatures and also at high air temperatures has a negative impact on cycling". So, making sure cyclists are comfortable during their journey is essential in encouraging them to keep cycling, even in these conditions. If the air temperature is high enough people will cycle less, taking other modes of transport. As cycling is a sustainable way of travel, this should be encouraged.
Next to cycling being a sustainable choice of transport, it is also a more healthy choice. According to the Dutch Heartfoundation, people who move sufficiently during the day, have 20 to 30 percent less chance of heart and vascular disease (De Nederlandse Hartstichting, n.d.). Sufficient movement is specified as 2 and a half hours per week of moderately intensive exercise, like cycling or walking. Therefore, encouraging cycling and discouraging sitting in a car or train are also a health benefit for everyone.

1.1. Goal Description

This paragraph will first describe some previous research, on which this thesis is based. Then, it will describe how this thesis will build on that previous research.

1.1.1. Building On Previous Research

This Bachelor Thesis is a continuation of a previous thesis by Steven Barendregt (2023). That thesis was about improving cycling conditions in high temperatures during summer. He researched what the optimal cycling conditions are and how trees can improve these conditions. After that research he used QGIS and Python to analyse what percentage of cycling paths in Delft are covered by shade during the averagely warmest day in summer. He did this by using GIS data of tree coverage in Delft. He calculated what shadow these trees give given their surface and height. To improve the percentage of shadow coverage, trees were manually placed in QGIS. Afterwards, he analysed how much the placement of those trees would improve shadow coverage.

More than 200 standard trees were placed with this non-automated approach. This meant the scope of the project was limited to only the main cycle paths. If all the smaller, less used cycle paths were also considered, the placement would take way too much time. Therefore, the main goal of this thesis is to automate this process. This would solve the tediousness, and also contribute to more thorough research into more than just the main cycling network.

1.1.2. Further Optimizations

Three more points of improvement are identified, but these are less essential than the previously mentioned goal, because these are more optimization issues. The main goal should be achieved before these optimizations are implemented as well.

The first of these is the calculation of shadows. Previously, shadow was calculated only to the north. This was simplified, but not the whole reality. The sun doesn't just shine from the north, but on a whole range of angles during the day. Also the height of the shadow was calculated on the ground surface, but this does not take into account the height of a cyclist.

Second, the cycling network could be more extensively researched if more cycle paths are considered. For now, just the high-end cycle network is used, because of the tedious manual process. If this tediousness is removed, maybe more of the less important parts of the cycle network could be investigated as well. This will be further explained in section 3.1.1.

The third optimization is the use of only one standard size tree to improve shadow coverage. It is not desirable to see the exact same type of tree all over the city. On the same street it is common to see the same type of tree, but every street will differ in some ways. Due to this being a non-essential optimization, this will be broadly investigated in the literature research.

1.2. Research Questions

Using the previously described goal of this thesis the main research question is:

What is the optimal way to automate and improve the process to provide more shade in high temperatures for cyclists in Delft?

To answer this research question, some sub-questions have been defined:

1. What methods of automation are available?
2. Which of these automation methods is most useful and what is the best way to use this method to achieve results?
3. Which optimizations can be made to the process to make the end result more realistic?

The first two sub-questions are the most essential part of the literature research part of this thesis. The majority of literature research will be dedicated to automation methods. A few methods will be identified and explained. Each method has its advantages and disadvantages. In the second sub-question, each of these methods will be explained. The most useful method chosen here will be used in the rest of the research to reach the goals set out.

The third sub-question is mainly focused on in the results of this thesis. It will focus on how the chosen automation method will be used to achieve the best result. It will be the most technical part of this thesis and will, no doubt, bring the most technical challenges.

The fourth sub-question will try to build on sub-question 3 as much as possible to implement the possible optimizations mentioned before. Some further literature studies could be necessary to find the best way to implement these optimizations.

2

Literature Research

This chapter will discuss gathered information necessary to answer the research questions. Section 2.1 will aim to answer the first sub-question. It will shine a light on available levels of automation and which tools are available for those levels. Later parts of section 2.1 will answer the second sub-question and will choose the optimal method for this thesis. Section 2.2 will describe the ways Dutch cities normally place new trees. This is essential to achieve a fairly realistic end result. Section 2.3 will discuss the research gap.

2.1. What methods of automation are available?

Multiple levels of automation in data science are available. The five levels of automation, described by Wang et al. (2021) are:

- **L0: No automation:** No automation, human performs the task.
- **L1: Human-directed automation:** System may provide automation functions (e.g., auto-sklearn), human selects methods and provide parameters before executing.
- **L2: System suggested, Human-directed:** System decides on parameters/methods/actions, human approves or changes suggested configuration before executing.
- **L3: System-directed automation:** System executes automatically and informs human with progress and results, human can intervene anytime.
- **L4: Full automation:** System decides everything and executes automatically, human is not in the loop.

A few of these are less useful for this thesis. For instance, level L0 is already present and is wished to be left behind for more automation. This level will, of course, not be considered in further parts of this report.

Secondly, level L3 is not achievable in the short span of this thesis. Also, it's not useful for the purposes the process is meant for. The system is meant to perform a few tasks and analyses. The human has to direct the computer to execute the process to achieve a relatively simple goal. No system-directed automation is necessary in this case. Therefore, level L4 is not applicable either. Here the human is not even in the loop, while it's the human who will request an answer in the end.

In the study by Wang et al. (2021) respondents involved in data science projects were asked what their current level of automation was, and what their desired level would be. This was done for 10 different stages of data science projects, including the 'Feature Engineering' phase. This phase resembles the goals of this thesis. Respondents involved in that phase told researchers, approximately, the distribution of current and desired levels of automation, shown at the top of page 4:

As seen in table 2.1, the most used and desired levels of automation are L1 and L2. For these reasons the most reasonable levels of automation, L1 and L2, will be discussed in further paragraphs.

2.1.1. L1: Human-directed automation

The first step from no automation to full automation is L1: Human-directed automation. The name suggests the way the system will automate the process. A human has to direct the system to make

Table 2.1: Approximates of levels of automation, according to Wang et al. (2021)

Level of automation	Current	Desired
L0	35%	0%
L1	35%	15%
L2	25%	50%
L3	5%	20%
L4	0%	15%

it perform certain automation tasks, mostly by providing parameters the system has to follow. So the human's task is to provide the right parameters to get the right result from the system.

In QGIS this can be done by using the model designer. In ArcGIS Pro this would be the ModelBuilder. These two features are almost identical in their use. What these two features make possible is the chaining of processing tools. In both QGIS and ArcGIS Pro processing tools have been made by the developers that can alter or analyse geodata. On their own, these tools are quite useful to do one particular task. When put in a model, a user can input a few parameters and make the system perform a chain of processing or analysis tools. This can automate a large number of tools, without the need to use every single one of them individually by hand. This model has the power to create new data from given data, and can use this new data to compare it to the old data. In the figures below, examples of QGIS and ArcGIS Pro are shown.

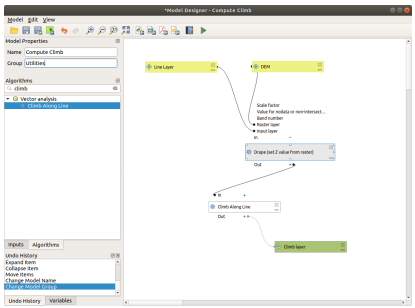


Figure 2.1: Example of the model designer in QGIS (QGIS Documentation, n.d.)

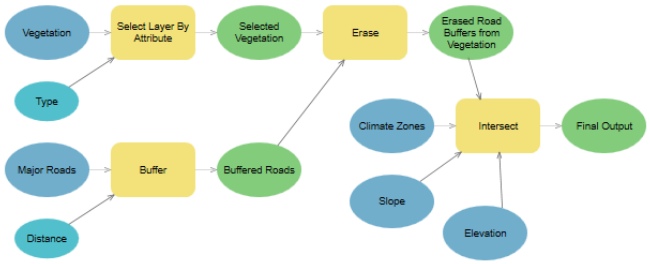


Figure 2.2: Example of the ModelBuilder in ArcGIS Pro (ArcGIS Pro | Documentation, n.d.)

The most important part about these models is entering the correct parameters. When working to a goal, through trial and error, the right combination and order of these processing tools can be identified. When done correctly, the model can be built according to the followed process, so that the model can duplicate the exact steps of the modeller. Afterwards, the system can be altered to new parameters to fit new goals. This is a great advantage to this method, since slightly different parameters don't mean having to restart the whole process.

Both QGIS and ArcGIS Pro have their own Python packages, namely PyQGIS and ArcPy. Both the model designer and ModelBuilder can be exported to these Python packages and can be used for further Python coding. This adds a lot of versatility for the user.

2.1.2. L2: System suggested, Human-directed

The next step to full automation is L2: System suggested, Human-directed. The name makes it seem like the system suggests certain direction the process could be taken. Afterwards, the human needs to direct the system on those suggestions. The human direction will remain largely the same, but some functionality has to be built upon that human direction. For the system to suggest certain things, it has to know what to look for and what type of things to suggest. These suggestions have to be based on something the system can figure out by itself. An example for this thesis would be like the following:

The system will identify which cycle paths could benefit most from the placement of new trees. After identifying a place, the user has to agree. Afterwards, the system will figure out what type of tree will be placed, which has to be confirmed by the user again. When all is agreed upon, the system will undergo its normal process and, when finished, present it to the user. If this is all satisfactory, the system will

save its result.

In this way, parameters can be figured out by the system, and the user just has to check if the process is being handled realistically. This is a process which is more useful, the more the process needs to be used. If the process needs to be run only a few times, this level of automation may not be necessary, as it would cost more time to reach this level of automation.

2.2. Trees around the city

As discussed in the introduction to this thesis, a discussion on the placement of trees around Dutch cities would be done. To make the most realistic dataset as possible, some knowledge about the placement of trees needs to be acquired.

According to a study by Smart et al. (2020), "...street tree density is defined as the number of trees per 100 m of street length...". This unit was used to research tree density in urban areas in multiple cities around the world. This unit is not the best estimator for shadow coverage, however. Not all trees are the same size and height. A street with a lot of small trees could get a high value of 20 trees per 100 m, but if these are very small, it could provide almost no shadow coverage. Also, if these trees are placed too far from the cycle paths, it could provide almost no shade. However, this unit could provide an indication of shade coverage, when taken into account the size of the trees.

A few streets in Delft, which provide good shadow coverage, were identified and the amount of tree density was noted, as well as the size of the trees and if both sides of the street have trees. This was done for a part of the street where the trees are continuously present, so no gaps are present. (Google, 2023). The table below shows these values for the identified streets.

Table 2.2: Tree density and tree size in Delft streets

Street name	Trees on both sides?	Tree density (trees per 100 m)	Tree size
Oostsingel	No	6.6	Large
Aan Het Verlaat	Yes	13.3	Medium
Kanaalweg	Yes	12.5	Large
Voorstraat (East side)	No	12.3	Small

As shown in Table 2.2, for bigger size trees, the amount of trees on one side of the street is about 6 to 7 trees per 100 m. For smaller trees, this is doubled to around 12 trees per 100 m. This is useful information, because the system has to be built to make a realistic end result. Therefore, it has to know how much distance should be in between trees. For instance, if 12 trees need to be placed every 100 meters, then one tree should be placed around every 8 meters.

2.3. Literature Gap

The literature research conducted in this thesis has left some gaps in the knowledge that is, therefore, missing in the end result. For instance, the way the shadows are calculated does not consider changes in the position of the sun. To make the end result more realistic, this factor needs to be taken into account.

Another point is that the trees are likely still not placed in a fully realistic manner. The municipality probably has some guidelines or, at least, common practices it uses to place new trees. To get a more realistic end result, this should be further investigated.

A point already raised by the previous thesis, is that the desired amount of tree coverage is not yet researched properly. This thesis only focuses on automating the process, but further research using this automation to reach a desired result should know how much shadow coverage should be aimed for. Some more research should be done on this, as the current assumption by Steven Barendregt is around 30% (Steven Barendregt, 2023).

3

Methodology

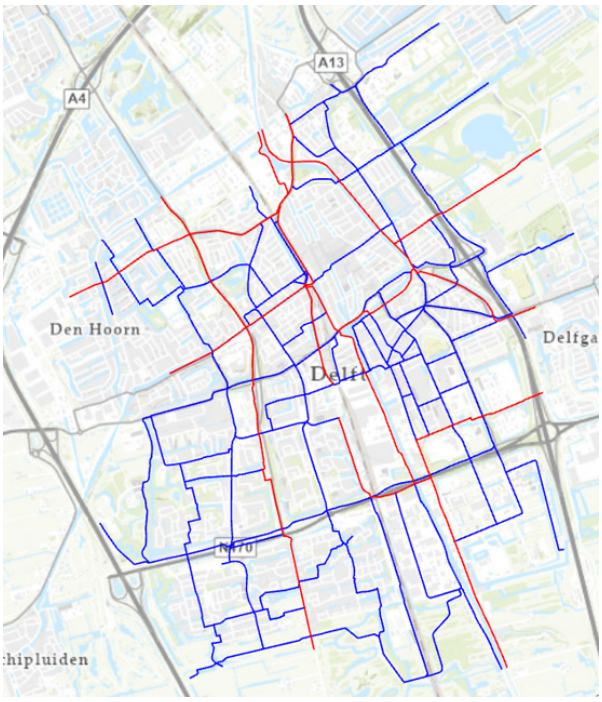
This chapter will describe the methods used to achieve the goals of this thesis. First, in section 3.1, the data and its origin will be explained. In section 3.2 the way the data will be processed and analysed is explained. This will include an explanation about which automation method is used.

3.1. Data Explanation

All data will be explained before moving further into the methodology. First, the cycling routes will be discussed, then the buildings and, lastly, data about trees will be explained.

3.1.1. Cycling Routes

In Steven Barendregt's thesis, he only looked at the high-end cycling network (Steven Barendregt, 2023). Data describing the more detailed main cycling network, was not used. It is available, however, and, can be taken into account for this thesis. The cycling routes are shown in Figure 3.1. The red routes are the high-end cycling routes. These will be looked at first. If the results are sufficient for these routes, then the main routes, shown in blue, can be looked at.



高端骑行网络
High-end cycling network
Detailed main cycling network

Figure 3.1: Cycling networks in Delft

The data was faulty at some points however. The cycling route data was reasonably altered for these clear faults. For instance, some routes were running through blocks of houses. One of these fixed errors is shown in Figure 3.2.
Another type of error were the existence of duplicates in both networks. These were identified to make sure all cycling paths only occur in one of the two data sets. Otherwise, some cycling paths will be analysed twice, which would give a small error in that analysis.
Also, some lines of only a few meters at the edge of the city were removed. These were too small to make any significant impact. Lastly, some lines were not connected logically to each other at a few places. Those small mistakes were fixed by making sure the lines were continuous again.

Figure 3.2: Example of an error in the cycling path data

Figure 3.1 already shows the data without these faults. This is the exact data that will be used in the later parts of this thesis.

3.1.2. Building Data

Some data provided by Steven Barendregt was not used in his thesis. This is true for the main cycling roads, as discussed before, but also for building data. This data describes all the buildings in Delft. It is shown in Figure 3.3. Firstly, this data can be used to know which sides of the road are available for trees. Secondly, it can be used to check if the process was done correctly. If trees are placed inside of a building, there has obviously gone something wrong. Buildings provide shade as well. This is not a part of this thesis. Only the shadow coverage from trees is considered. Making a full analysis on shadow coverage can be useful as well, but this thesis only looks at the contribution of the trees.

3.1.3. Tree Data

The tree data is divided in two parts, tree trunks and tree crowns. Both parts have different types of data inside them. The tree trunk data provides point data with information about the type of tree, the height, the thickness of the stem, etc. The tree crowns are just the surface of the tree, as polygon data. This polygon data, however, does not contain any information needed for the process. This data has to be attached from the tree trunk data to the tree crown data. Steven Barendregt already did this, as this is an essential step in the data processing. How this data will be attached is explained in the next section. The tree data is seen together with cycling routes and buildings in Figure 3.4.

Figure 3.3: Building data at Oostsingel

Figure 3.4: Tree data at Oostsingel

3.2. Data Processing

This section will explain in what ways the data will be altered. It will describe what will be done to reach the goals of this thesis. First, a flowchart will be given to show the steps in the process toward the end result. This flowchart gives a step by step list of processes the user has to take to achieve their goal. Secondly, every step in this flowchart will be explained. It will be explained how the user can follow this step, and also give some background on what this step achieves in the background. These steps are largely based on previous work (Steven Barendregt, 2023). When applicable, Steven Barendregt's process will be briefly explained to make clear how this thesis will build upon that work.

3.2.1. Flowchart

The flowchart to reaching the user's goal is given in Figure 3.5. Following these steps correctly is crucial to achieving the desired result. Each step of the flowchart is described briefly in Table 3.1. Each step is explained in more detail in the further parts of this section.

Table 3.1: Brief explanation of the flowchart per step

Step	Description
1: First shadowlength calculation	Calculating the shadowlength for the original tree data. This step only needs to be done once to get useful original data. Uses a model from the ModelBuilder named 'Original Shadowlength Calculation'
2: First analysis of shadow coverage	Analysing the original tree data to get a percentage value of already covered paths and a new layer of uncovered cycling paths. This new layer will be used to identify where new trees are needed. Uses a Python notebook named 'Original Shadow Coverage Notebook'
3: Choosing and preparing cycling paths to be covered	From the analysis in step 2, the user can see which cycling paths are not covered by shade. The user chooses paths they want to be covered and will do some preparatory work for those paths. Uses the model named 'Reversing Covered To Uncovered' and the Copy Parallel tool.
4: Placing trees along the prepared paths	The prepared paths will be used to place a certain type of tree along that path. This will result in new tree trunks and tree crowns. Uses a model from the ModelBuilder named 'Creating New Trees'
5: Second of analysis of shadow coverage	Using the same analysis method as in step 2, the percentage of new shadow coverage is calculated. Also a new layer of still uncovered paths is made. The user will then decide if the new coverage is enough. If this is not yet the case, the process is iterated from step 3. Uses a Python notebook named 'New Shadow Coverage Notebook'.

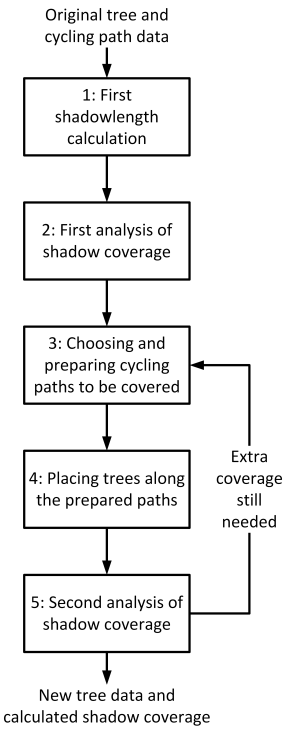


Figure 3.5: Flowchart for data creation and analysis

3.2.2. Step 1: First shadowlength calculation

Steven Barendregt already showed how to calculate the shadowlength from the tree data. Like explained before, some data needs to be attached to the tree crowns. This needs to be the shadowlength as described in Steven Barendregt's thesis (Steven Barendregt, 2023). Using the exact same calculations the length of the shadows will be known for every tree crown inside of the data. This shadowlength was based on an assumption about the position of the sun. For the purposes of automating the process to its fullest potential, this assumption will stay the same at first. Changing this assumption is not the highest priority for further optimizations. This assumption only has consequences for the length of the shadow, not the direction. The position of the sun has been previously assumed on the south. Therefore, the shadow was displaced to the north, by a distance of the shadowlength. Changing this direction would be an optimization with more priority than changing the shadowlength.

The equations Steven Barendregt used to calculate the shadowlength are as follows:

$$H_T = Trunkheight + \frac{(Treeheight - Trunkheight)}{2}$$

$$L_s = \frac{H_T}{tan(\alpha)}$$

With H_T = Height to middle of tree crown, L_s = Shadowlength & α = Angle of the sun.
This is still based on the same assumptions with the angle of the sun being 59.5 degrees (Steven Barendregt, 2023).

This step is achieved using a model built in the ModelBuilder in ArcGIS Pro. All the user will have to do is input the two original tree data sets, and the system will automatically create the desired result. This step only needs to be done once. After this, the result from step 1 will always be available to the user. The name of the model that has to be used is 'Original Shadowlength Calculation', shown in Appendix D.

3.2.3. Step 2: First analysis of shadow coverage

Steven Barendregt also showed how to calculate the shadow coverage (Steven Barendregt, 2023). This is done by shifting the tree crown data to the north by the length of the calculated shadowlength

from step 1. In step 2, the system will be checking where the calculated shadow surfaces are on top of the cycling paths. Then, the total length of covered cycling paths is compared to the total length of all cycling paths, to give a percentage of shadow coverage. The Python code Steven Barendregt used is shown in Appendix A.

This Python code needs to be made useful in a new notebook. The code used by Steven Barendregt can be inserted and adjusted to fit the goals of this thesis. This will give a notebook that can be run by the user. The user, then, just has to run the code to make the analysis. Only some small input parameters are needed, but the notebook will give the user suggestions on where to change the code to fit their goals, as well as giving examples of the inputs. The notebook is accessable to the user under the name 'Original Shadow Coverage Notebook' and is shown in Appendix B.

3.2.4. Step 3: Choosing and preparing cycling paths to be covered

From step 2, the user has gotten a new layer of cycling paths that are already covered by shadow. This step needs a layer of uncovered paths, however. So a small model has to be run to get the uncovered paths, instead of the covered ones. This model is called 'Reversing Covered To Uncovered', shown in Appendix E.

Looking at these paths will give the user some options where new trees could be beneficial. The user then has to select these paths and do some preparatory work to be able to move on to the next step. This preparatory work includes making a new line next to the existing path, using the Copy Parallel tool in ArcGIS Pro. This new line will be used as a guide to place trees along. This process is shown in Figure 3.6 & Figure 3.7.

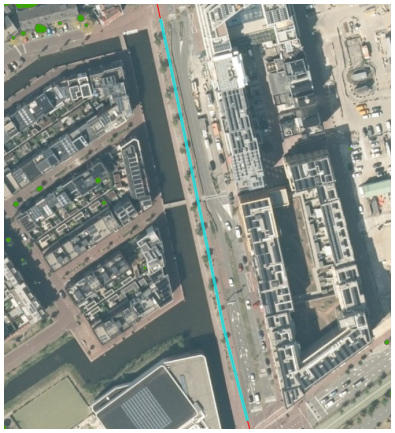


Figure 3.6: Example of a cycling path that was identified to benefit from new trees

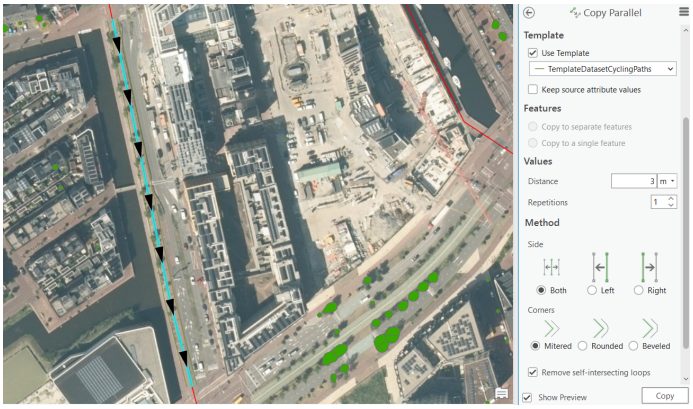


Figure 3.7: Example of how the Copy Parallel tool works

These two figures show an example of an isolated cycling path that was chosen to get more trees. As seen in the figures, the cycling path actually continues further to the north and south. The 'Split' tool was used in ArcGIS Pro. This tool requires the user to select a line and draw a new line to split the selected line at that point. This may have to be done to get a usable, isolated line. Figure 3.7 shows that this line will be copied 3 meters to the right and left. The amount of distance the line will be copied and to which side has to be chosen by the user themselves. For this example, both sides have room for new trees.

3.2.5. Step 4: Placing trees along the prepared paths

The next step in the process would be to use the prepared cycling paths to generate trees. This process is automated by using the ModelBuilder again. The model that is used in this step is named 'Creating New Trees' and is shown in Appendix F. This model will create new trees based on parameters the user will give. The user chooses the size of trees they want. In the background, this model places the trees first, and afterwards, runs the same process as in step 1. The end result of this step is, therefore, a layer with new trees that is identical in datatype as the trees from step 1. This new layer can then be used to complete the final step of the flowchart.

3.2.6. Step 5: Second analysis of shadow coverage

This step is identical to step 2. It only uses new data instead of the original data. A separate Python notebook is made for this step called 'New Shadow Coverage Notebook', shown in Appendix C. This notebook has some different suggestions from the first notebook to make sure the user doesn't mix up the data sets. The rest of the notebook does the exact same thing as the first notebook did. The outcome of this step a new layer of cycling paths that are still not covered with shadow. It also gives the additional percentage of shadow coverage given by the newly placed trees. Adding this new percentage to the percentage from the original data gives the percentage of shadow coverage from all trees. If this percentage is to the user's liking, the process can be stopped. If it it not yet up to the desired percentage, the process has to be repeated from step 3.

4

Results

This chapter will discuss all the results gathered from each step of the flowchart from Figure 3.5. In section 4.1, it gives a more detailed description of what each step achieves, by giving an example of how this thesis tested every method, and what the outcome of that testing was. Then, in section 4.2, it will show how to use the tools in every step by giving an example of the inputs necessary by the user.

4.1. Results

First, the results of every step are given using the outcomes of the testing for this thesis.

4.1.1. Step 1: First shadowlength calculation

The first step in the process is to calculate the shadowlength, which is needed for knowing how far to shift the shadows from the tree crown data. When these tree crowns are shifted by the shadowlength, the next step can be taken. For this step, the model in Appendix D is used. The end result of this step is two datasets. One of the tree trunks, and one of the tree crowns, which is identical to the tree trunks, only with the tree polygons, instead of just the points where the trunk is placed. In Figure 4.1 & Figure 4.2 the data fields before and after this step are shown for the tree crown data. What is seen is that all the data from the tree trunks is added, and the H_T and Shadowlength are added.

Current Layer: boomkronen

Visible	Read Only	Field Name	Alias	Data Type
☑	☐	uuid	uuid	Text
☑	☐	runnummer	runnummer	Text
☑	☐	id	id	Text
☑	☑	OBJECTID	OBJECTID	Object ID
☑	☐	SHAPE	SHAPE	Geometry

Figure 4.1: Data fields of the original tree crown data

Current Layer: Boomkronen Test

Visible	Read Only	Field Name	Alias	Data Type
☑	☐	uuid	uuid	Text
☑	☐	runnummer	runnummer	Text
☑	☐	id	id	Text
☑	☑	OBJECTID	OBJECTID	Object ID
☑	☐	SHAPE	SHAPE	Geometry
☑	☐	boomhoogte	boomhoogte	Double
☑	☐	opkroonhoogte	opkroonhoogte	Double
☑	☐	kroondiameter	kroondiameter	Double
☑	☐	kroonoppervlak	kroonoppervlak	Double
☑	☐	stamdiameter	stamdiameter	Short
☑	☐	kluitradius	kluitradius	Double
☑	☐	kluitdiepte	kluitdiepte	Double
☑	☐	boomsoort	boomsoort	Text
☑	☐	bag	bag	Short
☑	☐	water	water	Short
☑	☐	uid_gemeente	uid_gemeente	Text
☑	☐	H_T	H_T	Double
☑	☐	Shadowlength	Shadowlength	Double

Figure 4.2: Data fields of the modified tree crown data

4.1.2. Step 2: First Analysis of shadow coverage

Using a Python Notebook, shown in Appendix B, the shadow coverage by the original trees is obtained. This can now be done in eight directions. Not just the north, as was done by Steven Barendregt (Steven Barendregt, 2023), but also to all 7 other general directions, like south-west, for example. This gives the user more options to make sure that the shadow coverage is good for all times of the day. Doing this analysis for every direction gave eight separate percentages of shadow coverage, on both cycling networks. The notebook plots the tree data and cycling path data in multiple ways, and, at the end, gives the shadow coverage in a percentage for both cycling networks. These results are given in Table 4.1. As is seen, the main cycling network already has around 30% coverage in every direction. This means the high end cycling network benefits more from new trees.

Table 4.1: Shadow coverage per direction for both cycling networks

Direction	High end coverage	Main coverage
North	25.92%	29.65%
North-east	28.21%	30.13%
East	26.93%	30.36%
South-east	26.54%	30.52%
South	23.20%	31.36%
South-west	23.97%	31.93%
West	25.29%	31.31%
North-west	24.49%	29.89%

For further testing during this thesis, the south-west direction was used. This was chosen rather arbitrarily, as any direction would have been able to give new insights. The shadow coverage is shown in Figure 4.3 & Figure 4.4. The black lines, made extra thick for visualization purposes, in these figures are outputted by the notebook and were used in step 3. The notebook takes about three minutes to fully run.

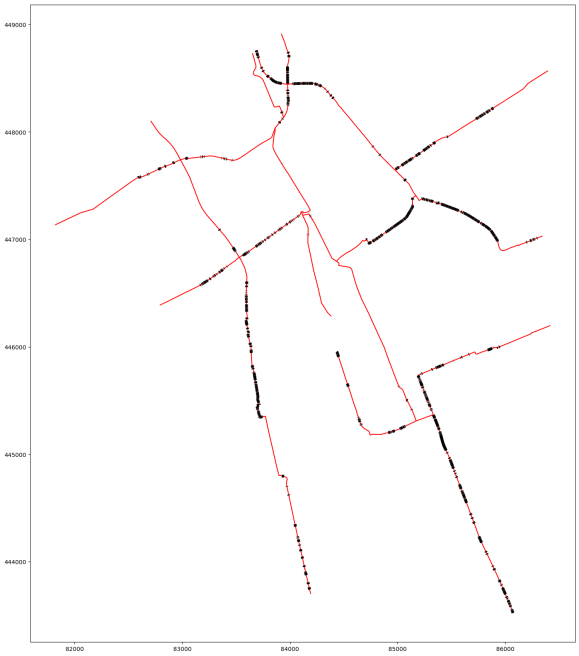


Figure 4.3: Shadow coverage from the original trees on the high end network

Figure 4.4: Shadow coverage from the original trees on the main network

4.1.3. Step 3: Choosing and preparing cycling paths to be covered

An example of how the cycling paths should be prepared was already given in Figure 3.6 & Figure 3.7. But before this can be done the model shown in Appendix E has to be run in ArcGIS Pro. The covered

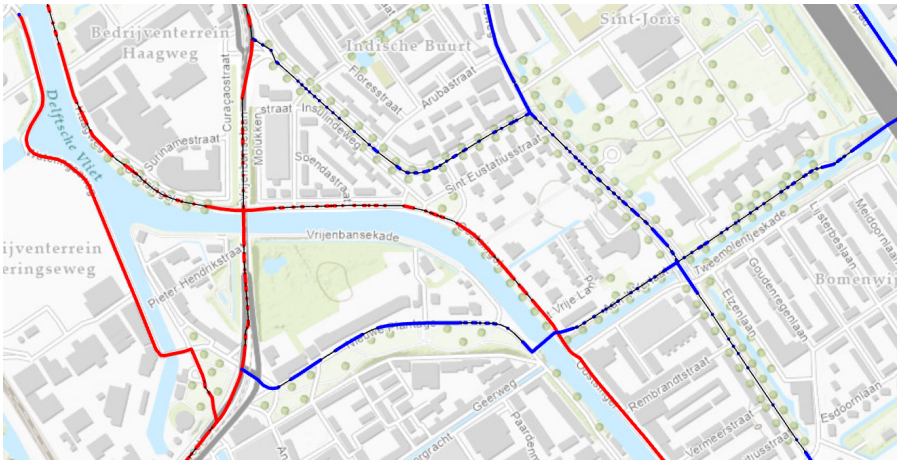


Figure 4.5: Example of the covered and uncovered lines cycling paths

paths are reversed to the uncovered paths using that model. The results are shown in Figure 4.5. The black lines are the covered paths, the uncovered lines are made thicker to make it more pronounced.

The process of choosing a cycling path and making a copy parallel to it is already shown in the Methodology chapter. The result of that part of the step is just the chosen and prepared cycling paths, shown in Figure 4.6.

Figure 4.6: Example of a prepared high end cycling path, shown in black

4.1.4. Step 4: Placing trees along the prepared paths

As already explained in section 3.2.5, the model called 'Creating New Trees', as shown in Appendix F is used for this step. In this model, the user can choose between four sizes of trees (small, medium, large and very large). Each of those sizes make trees with different parameters. There are four parameters which are different per size; the tree height, the tree crown height, the distance between trees and the

tree crown diameter. With the first two parameters, the same shadowlength calculation made in step 1 can be made for these new trees. The other two parameters are used for creating the tree crown polygons.

For the testing of this thesis large size trees are created. The new trees created by this model are shown in Figure 4.7. The trees do overlap, however, which would mean they may not be placed like this in real life. This is, however, not a negative thing for this thesis. Therefore, the trees are ready to be used in the last step.

Figure 4.7: Example of created large trees

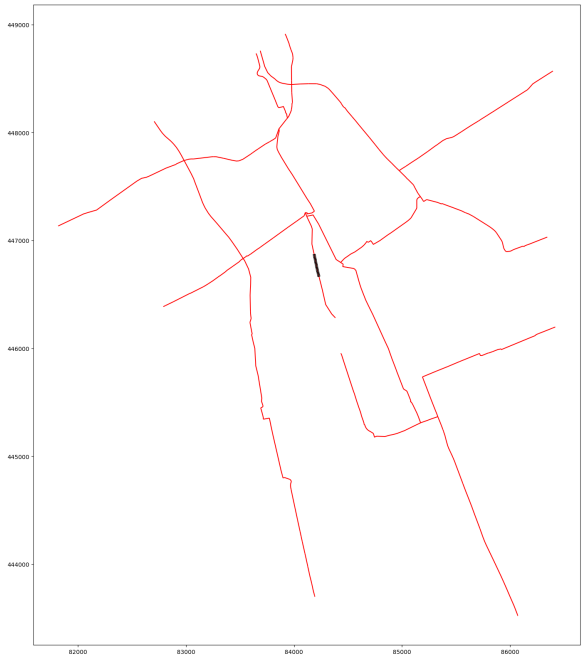


Figure 4.8: Shadow coverage from the new trees on the high end network

4.1.5. Step 5: Second analysis of shadow coverage

This step has exactly the same steps as step 2, using a slightly altered notebook, shown in Appendix C. The only difference is that the input file should be the newly created trees, instead of the original tree crowns. The notebook gives the same outputs as the first notebook. Thus, it also gives an additional shadow coverage percentage for the new trees. Since the newly created trees in this thesis only cover the high end cycling network, only a new percentage for that network is given. This percentage is 0.63%. The covered high end cycling paths are shown in Figure 4.8. The notebook only takes a few seconds to run, because only a few trees have to be loaded, in comparison to the around 100,000 trees in the original data.

4.2. User Manual

This section will give a user manual for every step. It will explain what inputs the user will have to give to reach a good result. It will also explain the example inputs given, which come from the testing for this thesis. The original data needed for the process is given in a .zip file. There, the user can find the original, unaltered, tree data and the cycling paths in a folder named 'Original Data'. The two notebooks are also provided in the .zip file. As a general rule, when a yellow triangle appears next to an input box in ArcGIS Pro, the model can still be run. Most of the time this means the output file already exists, and that the file will be overwritten. When a red symbol appears next to a box, this means the model can not be run. Most of the time this means that the input is not valid.

4.2.1. Step 1: First shadowlength calculation

Figure 4.9 shows the interface of the model that is used to calculate the shadowlengths. The first two boxes have to be filled with the original tree data, provided in the .zip file with all original data. The .zip file would have to be unpacked and the tree crown and tree trunk data have to be selected. The third box will have to be filled with the folder where the user want the output files to go. In this example, the folder name was 'Files'. A new user can make their own folder name, as long as that new folder is selected. The output file will be found in this output folder, to be used in further steps.

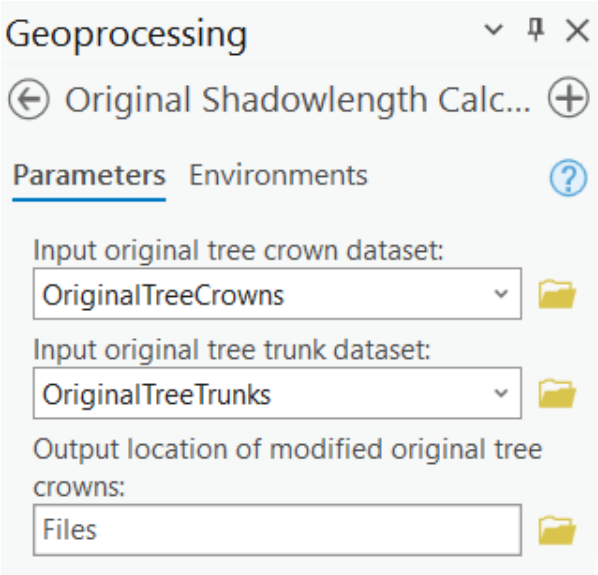


Figure 4.9: Interface of the 'Original Shadowlength Calculation' model

4.2.2. Step 2: First Analysis of shadow coverage

The notebook called 'Original Shadow Coverage Notebook', provided in the .zip file has to be opened in the user's preferred Python software, like Jupyter Notebook, for example. Make sure all necessary packages are installed. The packages are: pandas, geopandas and numpy, as well as all their required other packages.

```
In [1]: #Input your parameters here:
        OriginalTreeCrownData = "C:/Users/tieme/Desktop/BEP/Data/Files/OriginalTreeCrowns.shp"
        HighEndCyclingPaths = "C:/Users/tieme/Desktop/BEP/Data/Original Data/HighEndCyclingPaths.shp"
        MainCyclingPaths = "C:/Users/tieme/Desktop/BEP/Data/Original Data/MainCyclingPaths.shp"
        SunDirection = "SouthWest"
        OutputHighEndCyclingPaths = "C:/Users/tieme/Desktop/BEP/Data/Files/HighEndCoveredOriginally.shp"
        OutputMainCyclingPaths = "C:/Users/tieme/Desktop/BEP/Data/Files/MainCoveredOriginally.shp"
```

Figure 4.10: Interface of the cell where inputs have to be given in the 'Original Shadow Coverage Notebook'

At the beginning of the notebook, all inputs are asked to be filled in. This is shown in Figure 4.10. Before that, an explanation is already given on what the user should do in the notebook. The user has to input some files, for which the file path would have to be copied from the file explorer. Also, the direction the shadows should be calculated to has to be defined. Make sure to use the right spelling, as provided in the explanation at the top of the notebook. The last two lines in the cell give the output location of the covered cycling paths. An example of a name is given, and the same path for the folder as the output folder in step 1 is given. When running this notebook for multiple times, the user should be aware the output file name should be altered, because otherwise it would be overwritten.

4.2.3. Step 3: Choosing and preparing cycling paths to be covered

The first step in this step is the use of the 'Reversing Covered To Uncovered' model. Its interface is shown in Figure 4.11. Six inputs have to be given, all with an example present. The first two inputs are the original cycling path data, provided in the .zip file with original data. The second two inputs are the

output files from step 2. The last two inputs are the names and locations for the uncovered paths. In the example of the model, the same location is given as the covered cycling paths. The only difference is that the name now says 'Uncovered' instead of 'Covered'.

The next part of this step is already described in section 3.2.4. The user has to keep checking for themselves if what is being done is logical. The user can input the Building dataset, provided in the .zip file, or use satellite data, to check whether buildings are present. The best thing the user can do, is to make sure all cycling paths they want to include, end up in the same layer at the end. Also, the user should make sure the same boxes are checked as shown in Figure 3.7. The file given in 'Use Template', has to be a layer that is imported in ArcGIS Pro. This file is also given in the provided .zip file. Lastly, the user has to make sure the 'Distance' value is given in meters.

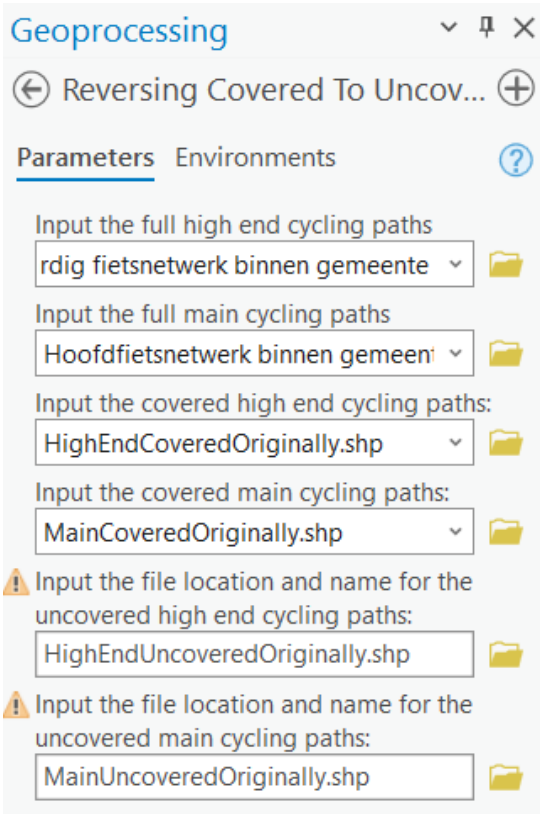


Figure 4.11: Interface of the 'Reversing Covered To Uncovered' model

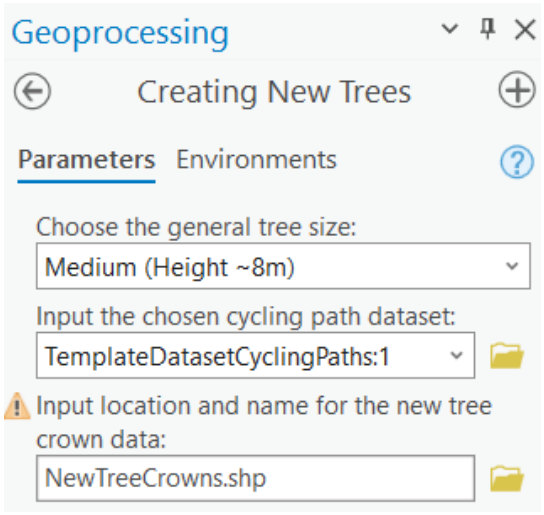


Figure 4.12: Interface of the 'Creating New Trees' model

4.2.4. Step 4: Placing trees along the prepared paths

This step uses the 'Creating New Trees' model. For now, this model gives all prepared cycling paths the same trees everywhere. In Figure 4.12, the interface of this model is shown. In the first box, the user has to input the desired tree size. The user has to make sure the chosen size is logical with the given cycling paths. The second box has to be filled with the prepared cycling paths. The last box has to be the location and name of the output file. This output file will be used in the final step.

4.2.5. Step 5: Second analysis of shadow coverage

This last step works almost exactly the same as step 2. The same code is used in the notebook, only the inputs are a little different, as seen in Figure 4.13. The first line should be the output file of step 4. The next two lines are the same as in the first notebook. They just input the cycling path data to be analysed. The next line is the direction again, and should be treated the same way as in the first notebook. It might be useful to keep this direction the same as in the first notebook, but the user could choose to analyse the newly created trees for every direction. In this way, the user knows what impact the new trees have at every part of the day. The last two files are the output locations. Again,

when running this for multiple times, the user should be aware the output file name should be altered, because otherwise it would be overwritten.

```
In [1]: #Input your parameters here:
        NewTreeCrownData = "C:/Users/tieme/Desktop/BEP/Data/Files/NewTreeCrowns.shp"
        HighEndCyclingPaths = "C:/Users/tieme/Desktop/BEP/Data/Original Data/HighEndCyclingPaths.shp"
        MainCyclingPaths = "C:/Users/tieme/Desktop/BEP/Data/Original Data/MainCyclingPaths.shp"
        SunDirection = "SouthWest"
        OutputHighEndCyclingPaths = "C:/Users/tieme/Desktop/BEP/Data/Files/HighEndCoveredNewly.shp"
        OutputMainCyclingPaths = "C:/Users/tieme/Desktop/BEP/Data/Files/MainCoveredNewly.shp"
```

Figure 4.13: Interface of the cell where inputs have to be given in the 'New Shadow Coverage Notebook'

5

Conclusion

The goal of this research was to create a more automated and realistic process for placing tree data and calculating shadow coverage from those trees on cycling paths. At the start of this research, some research was done into levels of automation. This was done to determine what level of automation was present before, and what level was possible to achieve for this research. From this, it was determined that the level 'L0: No automation' was present before, and that the next level, 'L1: Human-directed automation', was possible for this research. Further levels of automation would entail that the system will suggest changes itself, which is too difficult to achieve during this research, but also unnecessary, as the process would not be complicated enough to require that much automation. Next, some research was done into how far trees are placed apart in streets of Delft. A unit for the tree density was found for this, which means the amount of trees per 100 meters of street length. This tree density was later used to assume how far apart trees should be placed in the resulting models.

From the research into levels of automation, a method of automation was chosen. The use of Python notebooks and ArcGIS Pro models was determined to be the best way to move forward. These Python notebooks would be based on the code provided by Steven Barendregt (2023) in Appendix A and would calculate the shadow coverage. The models in ArcGIS Pro would be entirely new, but one of them had to be based on the previous work of Steven Barendregt as well. This would entail calculating the shadowlengths from the original tree data. The other two models would create the new tree data automatically, based on some small preparatory work the user will have to do.

In the end, the Python notebooks and ArcGIS Pro models will only need some input data to work. The user only has to make sure the inputs are done correctly. After that, the system will always do the same calculations. What each step of the system achieves is described in section 4.1. A user manual for each step in the process is given in section 4.2. This makes sure that new users will know what this process achieves, and how to achieve these results for themselves. In this way, anyone can pick up where this research left off, and improve even more on this process. In which ways the process can be improved is described in chapter 6.

6

Discussion & Recommendations

To make the results from this research even more realistic, some more research needs to be done. There are more than one way to improve in future research.

One way some big steps could be taken is to do more literature research. The spacing between trees and the size of these trees is based on some surface-level research. In the future, more research could be done on municipal standards on placement of trees or, at least, in which way trees are placed around cities. Therefore, making the tree data more realistic can be done in two ways. Firstly, an extensive research could be done into municipal or provincial standards into placement of trees. Secondly, a deep dive into actual tree density (amount of trees per 100 meters) around the city of interest. This would entail some fieldwork into mapping the tree density for a large amount of streets. Both of these methods would make the model more realistic, as the spacing and size of trees was assumed rather arbitrarily.

Another way to use literature research to achieve more realistic research is to find out how much shadow coverage is desired around an urban environment. This research was focused solely on the automation of the process, not on providing a data set of new trees with which the desired amount of shadow coverage is achieved, like the report of Steven Barendregt (2023). When this research is used to find out where to place new trees to achieve that result, some literature research is needed into how much shadow coverage is desired. This would the results more realistic and be useful for municipalities to know how to make their municipality more cyclist-friendly.

Another way to improve the results of this research is to improve the automated process. The process is now divided into two Python notebooks and three ArcGIS Pro models. This was done to make the process more clearly divided into steps, but this could all be combined into one Python notebook. As ArcGIS Pro allows models to be exported into Python code, which can only be used when using ArcGIS Pro's own Python environment, the notebooks and models could be combined into one notebook. The only thing that would make this hard is that the user still has to some manual preparatory work on choosing which cycling paths will be analysed. Therefore, combining it into two notebooks would be more realistic.

A very different way this research could be used is to use the shadow coverage calculation for determining the most shaded route in navigation apps. If these navigation apps, like Google Maps, know which cycling paths are covered in shade during high temperatures, the users of these apps could choose a more comfortable route. This could have an effect on how many people cycle on those days, opposed to taking their cars. For this, research has to be done how much the route can be diverted to make it more comfortable, but also make it acceptable for users.

This research could also be used to create new trees in different cities. The only things needed are new tree data and new cycling path data. Making sure this tree data is comparable to the data from this research is essential in this. The tree data should at least include the tree height and the height of

the tree crown. These two heights were used for the calculations and should, therefore, be present in the new data. In this research, the tree trunk data contained this information and the tree crowns only included the shapes. The models for calculating the shadowlength should be altered to fit the new data. The cycling path data should be line data. For Delft, two levels of the cycling network were present and used in the Python notebooks. If this is not the case in other cities, the notebooks should be altered to take this into account.

References

ArcGIS Pro | Documentation. (n.d.). *What is modelbuilder?* https://pro.arcgis.com/en/pro-app/latest/help/analysis/geoprocessing/modelbuilder/what-is-modelbuilder-.htm

De Kruijf, J., Van der Waerden, P., Feng, T., Böcker, L., Van Lierop, D., Ettema, D., & Dijst, M. (2021). Integrated weather effects on e-cycling in daily commuting: A longitudinal evaluation of weather effects on e-cycling in the netherlands. *Transportation Research Part A: Policy and Practice*, *148*, 305–315. https://doi.org/10.1016/j.tra.2021.04.003

De Nederlandse Hartstichting. (n.d.). *Lees meer over bewegen | hartstichting*. https://www.hartstichting.nl/gezond-leven/bewegen?gclid=CjwKCAjwvfmoBhAwEiwAG2tqzEuFio6lzOHtRKvP0VvJn3KUcfFP2fZYFppUM0fn58cROuyyFfyJHxoCb1cQAvD_BwE

Google. (2023). *Google Maps*. https://www.google.com/maps/

QGIS Documentation. (n.d.). *26.5. the model designer*. https://docs.qgis.org/3.28/en/docs/user_manual/processing/modeler.html

Smart, N., Eisenman, T. S., & Karvonen, A. (2020). Street tree density and distribution: An international analysis of five capital cities. *Frontiers in Ecology and Evolution*, *8*. https://doi.org/10.3389/fevo.2020.562646

Steven Barendregt. (2023). *Cycling In A Greener City*.

Wang, D., Liao, Q. V., Zhang, Y., Khurana, U., Samulowitz, H., Park, S., Muller, M., & Amini, L. (2021). How much automation does a data scientist want? https://doi.org/10.48550/arXiv.2101.03970

A

Steven Barendregt's Code

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import rasterio
import matplotlib
import os
import matplotlib.pyplot as plt
import contextily as cx
from rasterio.plot import show as rioshow
import geopandas as gpd
import pandas as pd
import glob



# In[2]:


files0 = glob.glob("Bep2.gpkg") #retrieving original data from QGIS
files1 = glob.glob("Bep2addition6.gpkg") #retrieving data with extra trees from QGIS


# In[3]:


gdf0 = gpd.read_file(files0[0]).set_crs('EPSG:28992') #loading tree data in python
gdf1 = gpd.read_file(files1[0]).set_crs('EPSG:28992') #loading extra tree data in python


# In[4]:


gdf0.head(5) #Shadowlength is different for each tree as is its surface


# In[5]:


gdf0.plot(figsize=(10,10), color='green') #map with the tree crown surface area


# In[ ]:
```

```
# In[6]:


#Shifting the surface of the tree crowns with the length of the shadow in the eastern
    direction
index = 0
gs0 = gpd.GeoSeries(data=gdf0.iloc[index].geometry)
output0 = gs0.translate(yoff=gdf0.iloc[index].shadowlength, xoff=0)

gs1 = gpd.GeoSeries(data=gdf1.iloc[index].geometry)
output1 = gs1.translate(yoff=gdf1.iloc[index].shadowlength, xoff=0)

fig, ax = plt.subplots(1)
gs0.plot(ax=ax,color="green")
output0.plot(ax=ax, color="black")


# In[7]:


lst_geoseries0 = []
for index in range(len(gdf0)):#Shifting for the whole original dataset
    gs0 = gpd.GeoSeries(data=gdf0.iloc[index].geometry)
    output0 = gs0.translate(yoff=gdf0.iloc[index].shadowlength, xoff=0)
    lst_geoseries0.append(output0)

lst_geoseries1 = []
for index in range(len(gdf1)):#Shifting for the whole dataset with extra trees
    gs1 = gpd.GeoSeries(data=gdf1.iloc[index].geometry)
    output1 = gs1.translate(yoff=gdf1.iloc[index].shadowlength, xoff=0)
    lst_geoseries1.append(output1)


# In[8]:


gs_combined0 = pd.concat(lst_geoseries0) #original
gdf_output0 = gpd.GeoDataFrame(geometry=gs_combined0)

gs_combined1 =pd.concat(lst_geoseries1) #with extra trees
gdf_output1 = gpd.GeoDataFrame(geometry=gs_combined1)


# In[9]:


gdf_output0.reset_index().drop(columns="index",inplace=True) #original
gdf_output1.reset_index().drop(columns="index",inplace=True) #with extra trees


# In[10]:


fig, ax = plt.subplots(1, figsize=(10,10))
gdf_output0.plot(ax=ax, color="black",zorder=-1)


# In[11]:


file_cycling = glob.glob("CyclingnetworkDelft.gpkg")
gdf_cycling = gpd.read_file(file_cycling[0]).set_crs('EPSG:28992') #loading cycling route
    data in python


# In[12]:


fig, ax = plt.subplots(1, figsize=(20,20)) #combining the maps of shadow and cycling routes
gdf_cycling.plot(ax=ax, color='red')
gdf_output0.plot(ax=ax, color="black",zorder=-1)
```

```
120
121
122 # In[13]:
123
124
125 merged0 = gdf_cycling.overlay(gdf_output0.set_crs('EPSG:28992'), how='intersection') #showing
        the places where cycling path are covered in shadow
126 merged0.plot(figsize=(10,10), color='purple')
127
128 s_total = gdf_cycling.length.sum()#length of cycling path
129 s0 = merged0.length.sum()#length of cycling path covered by shadow
130 print(s_total)
131 print(s0)
132 area0 = s0/s_total * 100 # percentage of cycling path covered in shadow
133 print(area0)
134
135
136 # In[14]:
137
138
139 fig, ax = plt.subplots(1, figsize=(20,20)) #showing were the shadow is lacking on the cycling
        paths in yellow
140 gdf_cycling.plot(ax=ax, color='yellow')
141 merged0.plot(ax=ax, color='purple')
142
143
144 # In[15]:
145
146
147 fig, ax = plt.subplots(1, figsize=(20,20)) #combining the maps of shadow and cycling routes
148 gdf_cycling.plot(ax=ax, color='red')
149 gdf_output1.plot(ax=ax, color="black",zorder=-1)
150
151
152 # In[16]:
153
154
155 merged1 = gdf_cycling.overlay(gdf_output1.set_crs('EPSG:28992'), how='intersection') #showing
        the places where cycling path are covered in shadow
156 merged1.plot(figsize=(10,10), color='purple')
157
158 s_total = gdf_cycling.length.sum()#length of cycling path
159 s1 = merged1.length.sum()#length of cycling path covered by shadow
160 print(s_total)
161 print(s1)
162 area1 = s1/s_total * 100 # percentage of cycling path covered in shadow
163 print(area1)
164
165
166 # In[17]:
167
168
169 fig, ax = plt.subplots(1, figsize=(20,20)) #showing were the shadow is lacking on the cycling
        paths in yellow
170 gdf_cycling.plot(ax=ax, color='yellow')
171 merged1.plot(ax=ax, color='purple')
172
173
174 # In[ ]:
175
176
177
178
179
180 # In[ ]:
```

B

Original Shadow Coverage Notebook

```
#!/usr/bin/env python
# coding: utf-8

# This notebook needs some input from the user to work. In the cell below, examples are given of what input is required.
#
# First, give the paths to all the files needed for this notebook. The three files are:
#     - Tree crown data with calculated shadowlengths
#     - High end cycling paths
#     - Main cycling paths
# The tree crown data path is the same path given to the first ArcGIS model for calculating the shadowlength.
#
# Second, the direction the shadow is shifted is needed. The eight choices are:
#     - North
#     - NorthEast
#     - East
#     - SouthEast
#     - South
#     - SouthWest
#     - West
#     - NorthWest
# Make sure the same spelling is used.
#
# The last thing the notebook needs is the path and names for the output files. In this case, these files would be the originally covered cycling paths for both levels.

# In[1]:


#Input your parameters here:
OriginalTreeCrownData = "C:/Users/tieme/Desktop/BEP/Data/Files/OriginalTreeCrowns.shp"
HighEndCyclingPaths = "C:/Users/tieme/Desktop/BEP/Data/Original Data/HighEndCyclingPaths.shp"
MainCyclingPaths = "C:/Users/tieme/Desktop/BEP/Data/Original Data/MainCyclingPaths.shp"
SunDirection = "SouthWest"
OutputHighEndCyclingPaths = "C:/Users/tieme/Desktop/BEP/Data/Files/HighEndCoveredOriginally.shp"
OutputMainCyclingPaths = "C:/Users/tieme/Desktop/BEP/Data/Files/MainCoveredOriginally.shp"


# In[2]:


import matplotlib
import matplotlib.pyplot as plt
import geopandas as gpd
import pandas as pd
import numpy as np
import os
get_ipython().run_line_magic('matplotlib', 'inline')
```

```

# In[3]:


gdf0 = gpd.read_file(OriginalTreeCrownData).set_crs('EPSG:28992')

gdf0.plot(figsize=(10,10), color='green') #map with the tree crown surface area


# In[4]:


#Shifting for the whole original dataset. Choose which side the shadow goes, north is the default.
lst_geoseries0 = []
for index in range(len(gdf0)):
    gs0 = gpd.GeoSeries(data=gdf0.iloc[index].geometry)
    if SunDirection == "North":
        output0 = gs0.translate(xoff=0, yoff=gdf0.iloc[index].Shadowleng)
    elif SunDirection == "NorthEast":
        output0 = gs0.translate(xoff=gdf0.iloc[index].Shadowleng*np.sqrt(2)/2, yoff=gdf0.iloc[index].Shadowleng*np.sqrt(2)/2)
    elif SunDirection == "East":
        output0 = gs0.translate(xoff=gdf0.iloc[index].Shadowleng, yoff=0)
    elif SunDirection == "SouthEast":
        output0 = gs0.translate(xoff=gdf0.iloc[index].Shadowleng*np.sqrt(2)/2, yoff=gdf0.iloc[index].Shadowleng*np.sqrt(2)/-2)
    elif SunDirection == "South":
        output0 = gs0.translate(xoff=0, yoff=-gdf0.iloc[index].Shadowleng)
    elif SunDirection == "SouthWest":
        output0 = gs0.translate(xoff=gdf0.iloc[index].Shadowleng*np.sqrt(2)/-2, yoff=gdf0.iloc[index].Shadowleng*np.sqrt(2)/-2)
    elif SunDirection == "West":
        output0 = gs0.translate(xoff=-gdf0.iloc[index].Shadowleng, yoff=0)
    elif SunDirection == "NorthWest":
        output0 = gs0.translate(xoff=gdf0.iloc[index].Shadowleng*np.sqrt(2)/-2, yoff=gdf0.iloc[index].Shadowleng*np.sqrt(2)/2)

    lst_geoseries0.append(output0)

gs_combined0 = pd.concat(lst_geoseries0)
gdf_output0 = gpd.GeoDataFrame(geometry=gs_combined0)
gdf_output0.reset_index().drop(columns="index",inplace=True)


# In[5]:


fig, ax = plt.subplots(1, figsize=(10,10))
gdf_output0.plot(ax=ax, color="black",zorder=-1)


# In[6]:


#loading high end cycling paths data in python. Input the given original data
gdf_cycling1 = gpd.read_file(HighEndCyclingPaths).set_crs('EPSG:28992')


# In[7]:


#combining the maps of shadow and high end cycling routes
fig, ax = plt.subplots(1, figsize=(20,20))
gdf_cycling1.plot(ax=ax, color='red')
gdf_output0.plot(ax=ax, color="black",zorder=-1)


# In[8]:

```

```

#showing the places where high end cycling paths are covered in shadow
merged1 = gdf_cycling1.overlay(gdf_output0.set_crs('EPSG:28992'), how='intersection')
merged1.plot(figsize=(10,10), color='purple')

s_total = gdf_cycling1.length.sum() #length of high end cycling path
s1 = merged1.length.sum() #length of high end cycling path covered by shadow
area1 = round(s1/s_total * 100, 2) # percentage of high end cycling path covered in shadow


# In[9]:


#showing were the shadow is lacking on the high end cycling paths in yellow
fig, ax = plt.subplots(1, figsize=(20,20))
gdf_cycling1.plot(ax=ax, color='red')
merged1.plot(ax=ax, color='black')


# In[10]:


#loading main cycling paths data in python. Input the given original data
gdf_cycling2 = gpd.read_file(MainCyclingPaths).set_crs('EPSG:28992')


# In[11]:


#combining the maps of shadow and main cycling routes
fig, ax = plt.subplots(1, figsize=(20,20))
gdf_cycling2.plot(ax=ax, color='blue')
gdf_output0.plot(ax=ax, color="black",zorder=-1)


# In[12]:


#showing the places where main cycling paths are covered in shadow
merged2 = gdf_cycling2.overlay(gdf_output0.set_crs('EPSG:28992'), how='intersection')
merged2.plot(figsize=(10,10), color='purple')

s_total = gdf_cycling2.length.sum() #length of main cycling path
s2 = merged2.length.sum() #length of main cycling path covered by shadow
area2 = round(s2/s_total * 100, 2) # percentage of main cycling path covered in shadow


# In[13]:


#showing were the shadow is lacking on the main cycling paths in yellow
fig, ax = plt.subplots(1, figsize=(20,20))
gdf_cycling2.plot(ax=ax, color='blue')
merged2.plot(ax=ax, color='black')


# In[14]:


#Choose where to export the shapefiles for the high end and main cycling paths. Optionally
    give the direction used: North, south, etc.
merged1.to_file(OutputHighEndCyclingPaths, crs = 'EPSG:28992')
merged2.to_file(OutputMainCyclingPaths, crs = 'EPSG:28992')


# In[15]:


print("Percentage of high end cycling paths covered by shadow originally:", area1,"%")
print("Percentage of main cycling paths covered by shadow originally:", area2,"%")
```

C

New Shadow Coverage Notebook

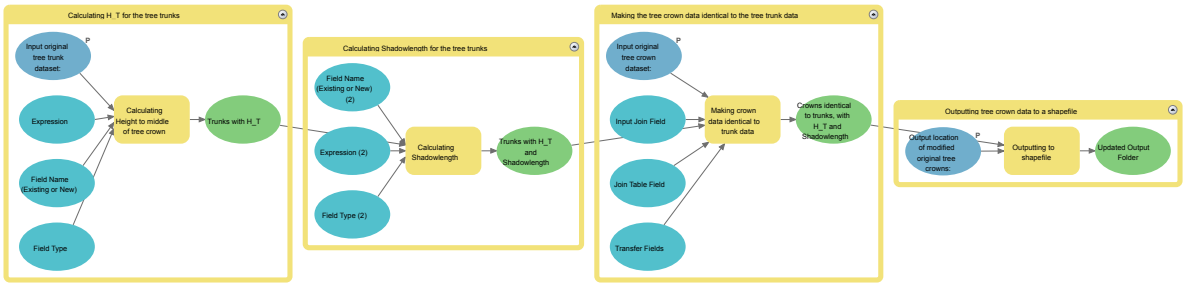
```
#!/usr/bin/env python
# coding: utf-8

# This notebook needs some input from the user to work. In the cell below, examples are given of what input is required.
#
# First, give the paths to all the files needed for this notebook. The three files are:
#     - Tree crown data with calculated shadowlengths
#     - High end cycling paths
#     - Main cycling paths
# The tree crown data path is the same path given to the ArcGIS model creating the new trees.
#
# Second, the direction the shadow is shifted is needed. The eight choices are:
#     - North
#     - NorthEast
#     - East
#     - SouthEast
#     - South
#     - SouthWest
#     - West
#     - NorthWest
# Make sure the same spelling is used.
#
# The last thing the notebook needs is the path and names for the output files. In this case, these files would be the newly covered cycling paths for both levels.

# In[1]:


#Input your parameters here:
NewTreeCrownData = "C:/Users/tieme/Desktop/BEP/Data/Files/NewTreeCrowns.shp"
HighEndCyclingPaths = "C:/Users/tieme/Desktop/BEP/Data/Original␣Data/HighEndCyclingPaths.shp"
MainCyclingPaths = "C:/Users/tieme/Desktop/BEP/Data/Original␣Data/MainCyclingPaths.shp"
SunDirection = "SouthWest"
OutputHighEndCyclingPaths = "C:/Users/tieme/Desktop/BEP/Data/Files/HighEndCoveredNewly.shp"
OutputMainCyclingPaths = "C:/Users/tieme/Desktop/BEP/Data/Files/MainCoveredNewly.shp"


# In[2]:


import matplotlib
import matplotlib.pyplot as plt
import geopandas as gpd
import pandas as pd
import numpy as np
import os
get_ipython().run_line_magic('matplotlib', 'inline')


```

```
# In[3]:


gdf0 = gpd.read_file(NewTreeCrownData).set_crs('EPSG:28992')

gdf0.plot(figsize=(10,10), color='green') #map with the tree crown surface area


# In[4]:


lst_geoseries0 = []
for index in range(len(gdf0)):
    gs0 = gpd.GeoSeries(data=gdf0.iloc[index].geometry)
    if SunDirection == "North":
        output0 = gs0.translate(xoff=0, yoff=gdf0.iloc[index].Shadowleng)
    elif SunDirection == "NorthEast":
        output0 = gs0.translate(xoff=gdf0.iloc[index].Shadowleng*np.sqrt(2)/2, yoff=gdf0.iloc
            [index].Shadowleng*np.sqrt(2)/2)
    elif SunDirection == "East":
        output0 = gs0.translate(xoff=gdf0.iloc[index].Shadowleng, yoff=0)
    elif SunDirection == "SouthEast":
        output0 = gs0.translate(xoff=gdf0.iloc[index].Shadowleng*np.sqrt(2)/2, yoff=gdf0.iloc
            [index].Shadowleng*np.sqrt(2)/-2)
    elif SunDirection == "South":
        output0 = gs0.translate(xoff=0, yoff=-gdf0.iloc[index].Shadowleng)
    elif SunDirection == "SouthWest":
        output0 = gs0.translate(xoff=gdf0.iloc[index].Shadowleng*np.sqrt(2)/-2, yoff=gdf0.
            iloc[index].Shadowleng*np.sqrt(2)/-2)
    elif SunDirection == "West":
        output0 = gs0.translate(xoff=-gdf0.iloc[index].Shadowleng, yoff=0)
    elif SunDirection == "NorthWest":
        output0 = gs0.translate(xoff=gdf0.iloc[index].Shadowleng*np.sqrt(2)/-2, yoff=gdf0.
            iloc[index].Shadowleng*np.sqrt(2)/2)

    lst_geoseries0.append(output0)

gs_combined0 = pd.concat(lst_geoseries0)
gdf_output0 = gpd.GeoDataFrame(geometry=gs_combined0)
gdf_output0.reset_index().drop(columns="index",inplace=True)


# In[5]:


fig, ax = plt.subplots(1, figsize=(10,10))
gdf_output0.plot(ax=ax, color="black",zorder=-1)


# In[6]:


#loading high end cycling paths data in python. Input the given original data
gdf_cycling1 = gpd.read_file(HighEndCyclingPaths).set_crs('EPSG:28992')


# In[7]:


#combining the maps of shadow and high end cycling routes
fig, ax = plt.subplots(1, figsize=(20,20))
gdf_cycling1.plot(ax=ax, color='red')
gdf_output0.plot(ax=ax, color="black",zorder=-1)


# In[8]:


#showing the places where high end cycling paths are covered in shadow
merged1 = gdf_cycling1.overlay(gdf_output0.set_crs('EPSG:28992'), how='intersection')
merged1.plot(figsize=(10,10), color='purple')
```

```

s_total = gdf_cycling1.length.sum() #length of high end cycling path
s1 = merged1.length.sum() #length of high end cycling path covered by shadow
area1 = round(s1/s_total * 100, 2) # percentage of high end cycling path covered in shadow


# In[9]:


#showing were the shadow is lacking on the high end cycling paths in yellow
fig, ax = plt.subplots(1, figsize=(20,20))
gdf_cycling1.plot(ax=ax, color='red')
merged1.plot(ax=ax, color='black')


# In[10]:


#loading main cycling paths data in python. Input the given original data
gdf_cycling2 = gpd.read_file(MainCyclingPaths).set_crs('EPSG:28992')


# In[11]:


#combining the maps of shadow and main cycling routes
fig, ax = plt.subplots(1, figsize=(20,20))
gdf_cycling2.plot(ax=ax, color='blue')
gdf_output0.plot(ax=ax, color="black",zorder=-1)


# In[12]:


#showing the places where main cycling paths are covered in shadow
merged2 = gdf_cycling2.overlay(gdf_output0.set_crs('EPSG:28992'), how='intersection')
merged2.plot(figsize=(10,10), color='purple')

s_total = gdf_cycling2.length.sum() #length of main cycling path
s2 = merged2.length.sum() #length of main cycling path covered by shadow
area2 = round(s2/s_total * 100, 2) # percentage of main cycling path covered in shadow


# In[13]:


#showing were the shadow is lacking on the main cycling paths in yellow
fig, ax = plt.subplots(1, figsize=(20,20))
gdf_cycling2.plot(ax=ax, color='blue')
merged2.plot(ax=ax, color='black')


# In[14]:


#Choose where to export the shapefiles for the high end and main cycling paths. Optionally
    give the direction used: North, south, etc.
merged1.to_file(OutputHighEndCyclingPaths, crs = 'EPSG:28992')
merged2.to_file(OutputMainCyclingPaths, crs = 'EPSG:28992')


# In[15]:


print("Percentage␣of␣high␣end␣cycling␣paths␣covered␣by␣shadow␣newly:", area1,"%")
print("Percentage␣of␣main␣cycling␣paths␣covered␣by␣shadow␣newly:", area2,"%")
```

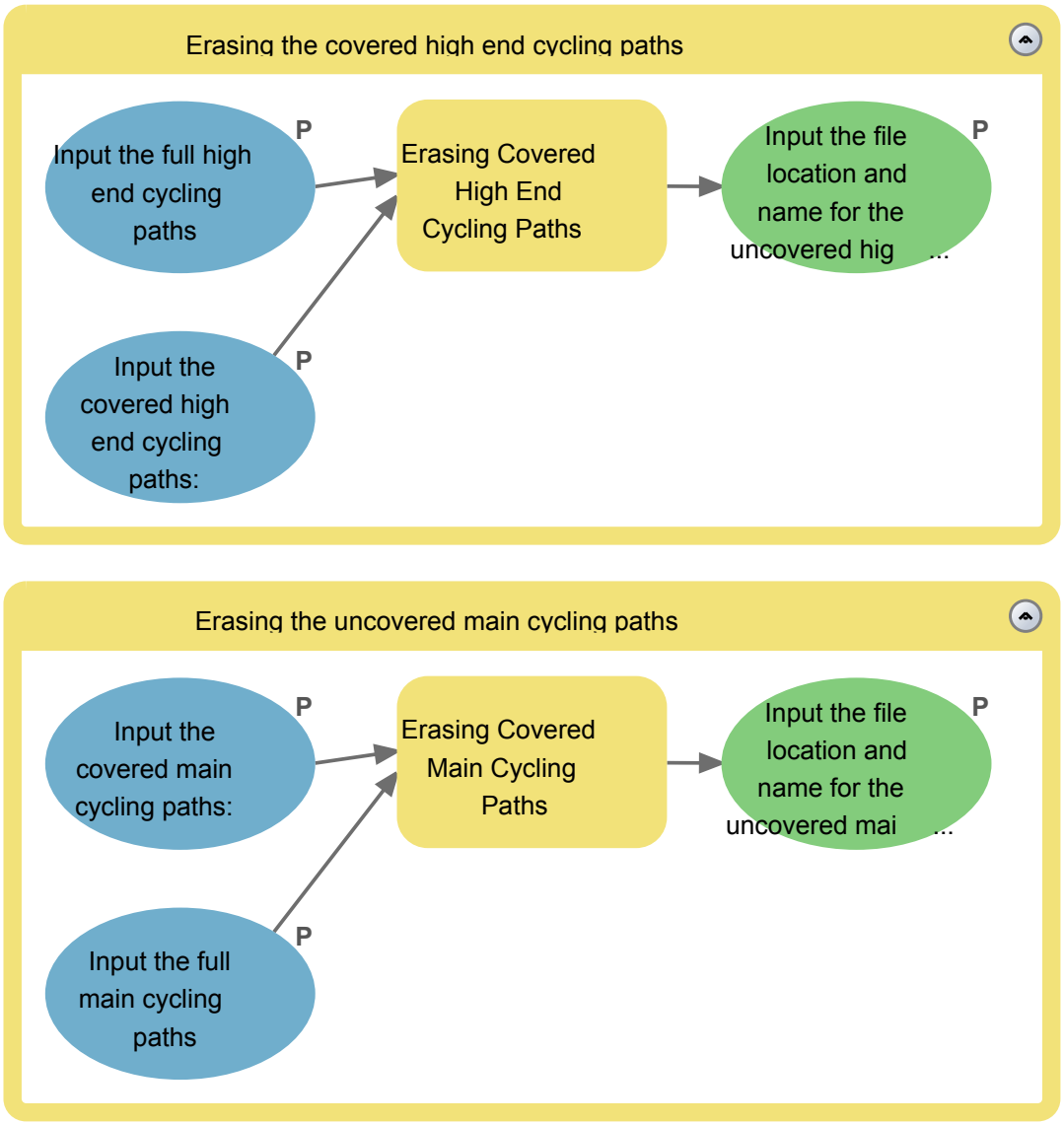
D

Original Shadowlength Calculation Model



E

Reversing Covered To Uncovered Model



F

Creating New Trees Model