Technische Universiteit
Eindhoven
University of Technology

Department of Mechanical Engineering

Representing obstacles using isolines with application to optimal motion planning

Bachelor Final Project
DC 2017.026

L. van Wincoop 0849855

Supervisors:
Dr. A. Saccon
B.A.C. van de Schoot BSc.

Eindhoven, February 3, 2017

Contents

1 Introduction

Motion planning is an important aspect of robot movement. Robotic arms use motion planning to grip objects and mobile robots use motion planning to determine paths along which they move.

Robot movement can be determined by motion planning algorithms. These algorithms need information about an approximation of the Signed Distance Function (SDF) from the robot to an obstacle. An SDF is a function that determines the distance of a certain point to the contour whose points form a set. If the point lies within the contour, it has a positive sign and if the point lies outside the contour it has a negative sign [1]. This type of information can originate from camera data for example. The observed obstacles often form a problem for these algorithms. In recent studies, the zero level sets of implicit functions have been used to store a description of the contours of the objects on grid data. This type of method is called a volumetric method.

Robots obtain information about obstacles either from camera data or models of the environment. One way to generate 3D models of the environment is the KinectFusion system [4], with data originating from the depth camera Kinect. The objects are represented by using the volumetric methods presented by Curless and Levoy in [3]. According to this paper, range images by depth cameras are used to determine SDFs that are stored on a voxel grid. The zero crossings in these SDFs are combined to form an implicit function which represents the surface of the objects. The signed distance values outside and inside the objects are truncated so that they form a Truncated Signed Distance Function (TSDF). This means that at increasing distance from the object surface, the signed distance values converge toward a constant value, for example -1. The same holds for the values inside the object but then for the positive value 1.

There are current motion planning algorithms that make use of SDFs and volumetric methods. One such algorithm is Covariant Hamiltonian optimization for motion planning (CHOMP) as presented by Zucker et. al in [9]. This algorithm uses functional gradient optimization. In other words, they minimize a functional which is a function of a set of other functions by using the gradient descent optimization method. The functional is a function of a set of functions, in this case trajectories. The trajectories are functions of cost elements and the goal is to find the trajectory that provides as low a cost for the robot as possible. A limiting factor in this algorithm is that the elements of the trajectory function are not twice continuously differentiable. This excludes the use of Newton-like optimization methods, which provide faster local convergence than the gradient method.

For this project, the goal is to explore the applicability of volumetric methods on Newton-like algorithms that can solve constrained optimization problem. This requires the volumetric function to be twice continuously differentiable.

Te report has the following structure. Chapter 2 consists of two parts. The first part shows the grid data that is synthetically created such that it can represent the type of data one would obtain from using volumetric methods. After this, the method to render the grid data twice continuously differentiable is presented. This means ensuring that the grid data is twice continuously differentiable. Next, the Newton-like algorithm is created. In Chapter 3, the application of the volumetric method on the Newton-like algorithm is validated. This validation mainly consists of checking the convergence and rate of convergence for different obstacle shapes. Lastly, Chapter 4 presents the conclusions and recommendations that are based on the results from the validation.

2 Problem Statement and Approach

In this chapter, the problem statement will be outlined. After this, the synthetization of the grid data is shown and consequently the interpolation of the data to provide a twice continuously differentiable approximation is presented. Lastly, the Newton-like optimization algorithm is described.

2.1 Problem statement

In order to assess if the volumetric methods can be used in Newton-like methods, the grid needs to be synthesized such that it represents a grid with a truncated signed distance function and the zero level set of an object. The next part of the problem is making sure that the grid values are twice continuously differentiable.

When the data is made twice continuously differentiable, a Newton-like algorithm needs to be customized that can be applied on a constrained optimization problem. The optimization problem used is the minimization of the distance toward a certain point within the contour of an object on the grid, while remaining outside the contour.

As a simplificiation the choice is made to only consider the 2D case. The goal of this project is to validate if applying volumetric methods on Newton-like algorithms is possible. Only considering the 2D situation, is sufficient to reach this goal. In later research, the extension to 3D needs to be made in order to apply the algorithm for motion planning.

2.2 Grid generation and interpolation

The grid consists of two coordinates $x = [x_1, x_2]^T$.

An ellipse is used as the object on the grid. The implicit function that gives the level sets of the ellipse is

$$u = \frac{x_1^2}{0.5} + \frac{x_2^2}{0.2} - 1. \tag{2.1}$$

The parameters 0.2 and 0.5 determine the size of the ellipse. They are chosen as 0.2 and 0.5 since this ellipse fit the grid well. The value of the implicit function at each grid point represents the signed distance value at that point. Figure 2.1 shows the zero level set of the ellipse and illustrates the position of the ellipse on the grid. This contour has been created using the marching squares algorithm, which is a computer algorithm that is used to plot the level set curves of a certain shape. In this case the zero level set curve of the ellipse was plotted. [5]

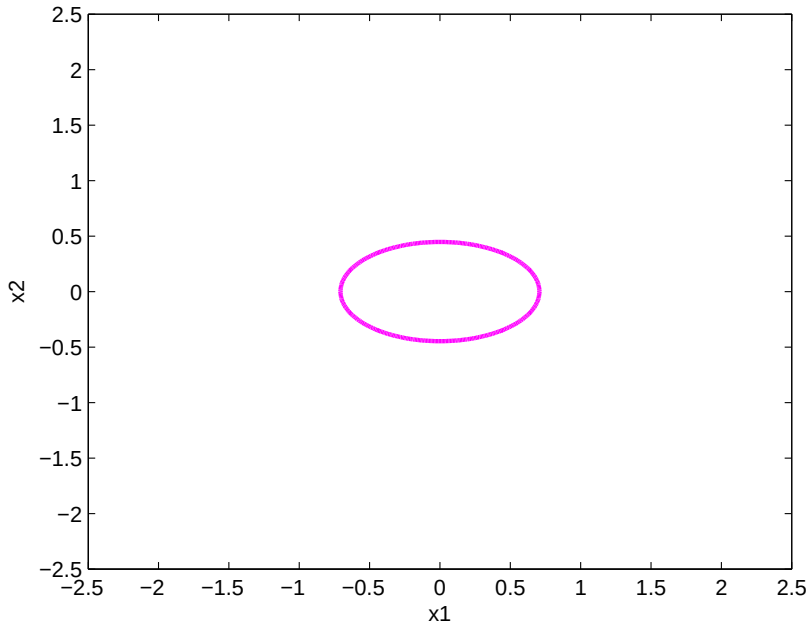


Figure 2.1: The ellipse on the grid

A distance of 0.01 between each grid point has been chosen and each of the grid coordinates has a lower bound of -2.5 and an upper bound of 2.5. This results in a total of 251001 data points.

In order to truncate the grid data to represent the effect of a TSDF, (2.1) is saturated between -1 and 1 by

$$q = -\frac{2}{1+e^{-2u}} + 1. \tag{2.2}$$

This equation is an altered version of the Sigmoidal curve or the Logistic function [8]. By adjusting the parameters in the numerator and denominator, one can change the steepness of the curve and its maximum and minimum value. By substituting Equation (2.1) to (2.2), one obtains

$$q = \frac{2}{1+e^{-2\left(\frac{x_1^2}{0.5}+\frac{x_2^2}{0.2}-1\right)}}. \tag{2.3}$$

Now that the function values on the grid have been generated, these values need to be interpolated such that they are twice continuously differentiable. Then they can be applied to the Newton-like optimization algorithm. The requirement is that the interpolant that results is twice continuously differentiable. A cubic spline consists of piecewise third-order polynomials. These polynomials are defined on the intervals between grid points. Boundary conditions are applied in these points to realise smooth transitions between the polynomials, which results in a twice continuously differentiable spline [2]. Therefore, cubic splines should meet the requirement. The points of the interpolant are now denoted by $x = [x_1, x_2]^T$, where x can take on all real values between -2.5 and 2.5. In Figure 2.2, the resulting spline is shown for the truncated grid values.

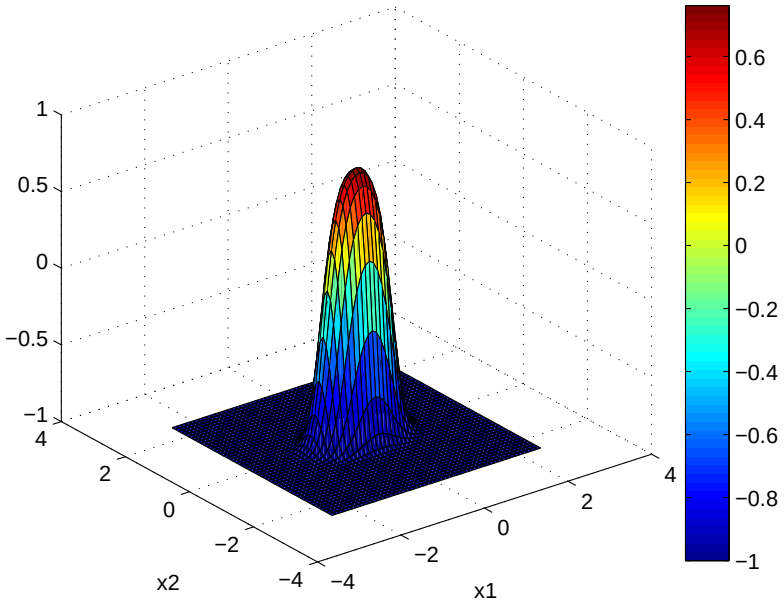


Figure 2.2: Cubic spline interpolation of the grid values

2.3 Optimization algorithm

Now that the grid has been defined and interpolated to form a twice continuously differentiable function, the optimization algorithm can be presented. The goal is to minimize the distance toward a certain point within the boundary of the ellipse, without entering the object itself. The unconstrained function is shown in

$$f(x) = (x - x_c)^T (x - x_c), \tag{2.4}$$

where $x_c = [a, b]^T$ is the unconstrained minimizer x_f^*. The minimum of the function $f(x)$ is indicated by $f(x_f^*)$. The minimizer x_f^* is chosen to lie within the contour of the ellipse namely, at the point $x_f^* = [0, 0]^T$. Minimizing $f(x)$ would therefore cause the algorithm to reach a point within the ellipse. For this reason, $f(x)$ should be minimized with extra condition that $c < 0$, where c is the value of the interpolated data in each point x.

The constrained optimization problem is generated by introducing a logarithmic-barrier function, which is real-valued outside the contours of the ellipse. The logarithmic-barrier function belongs to an optimization method called the interior point method [6]. When constraining the optimization problem, the following equation is obtained:

$$h(x) = f(x) - \epsilon \, log(-c), \tag{2.5}$$

where ϵ is the barrier parameter, which is a positive scalar, $h(x)$ is called the objective function, which is the function to be minimized. As the barrier parameter approaches zero, the objective function should converge to a minimizer close to the unconstrained minimizer without crossing the barrier. This means that for $\epsilon \to 0$, the minimizer will lie on the contour of the ellipse. Because of the positive values for c within the contours of the obstacle and negative values outside the obstacle, the log-term is real-valued outside the object and inside the boundary the values become imaginary with an asymptote at the contour [6]. For illustration purposes, the one-dimensional cases of (2.4) and (2.5) are shown in Figure 2.3.

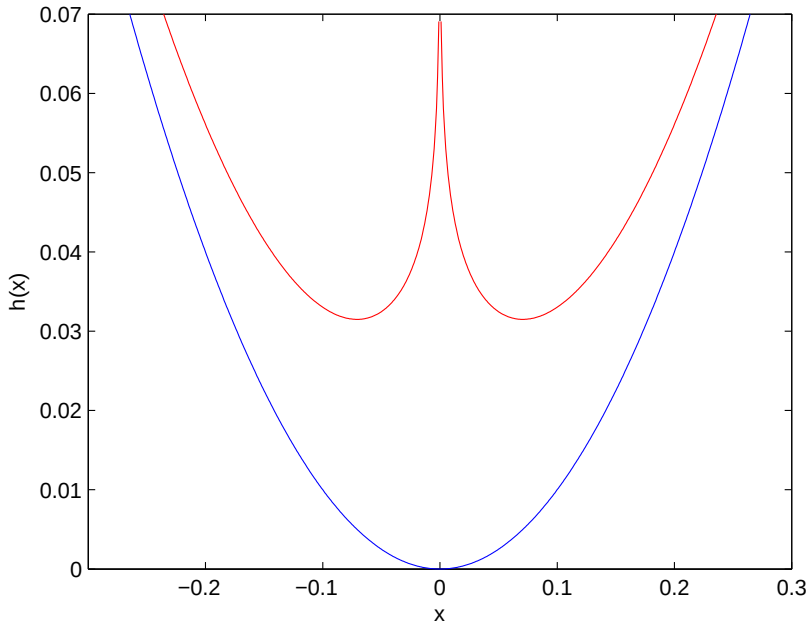


Figure 2.3: Illustration of the influence of the barrier-function for the one-dimensional case

This figure shows the effect of adding the log-barrier. The blue graph shows the one dimensional version of function $f(x)$ with the unconstrained minimizer at $x_f^* = 0$ and the red graph shows what happens to $f(x)$ when adding the barrier term. The function now has two minimizers on either side of $x = 0$.

In the following part the derivations shown are based on theory from [7]. For this algorithm, the Newton method is preferred over the gradient descent method. The difference between the two methods is that the steepest descent method, or gradient method, uses a local linear approximation and the Newton method uses a local second order approximation. This makes the steepest descent method mathematically less complex since the function only needs to be once continuously differentiable. In order to calculate the next point x to be approximated, the steepest descent method uses the gradient of the objective function, multiplied by a factor as shown in

$$x_{i+1} = x_i - \alpha_i Dh(x_i) \tag{2.6}$$

With i denoting the iteration number, $i \in \mathbb{N}$. α_i is the step size and $Dh(x_i)$ is the derivative of the objective function $h(x_i)$. When the algorithm approaches a minimum, the gradient will have very small values and therefore, the difference between x_i and x_{i+1} will be very small. This means that the progress toward the minimum will be very slow close to the minimizer. The Newton method provides a solution for this by using second order terms that ensure fast convergence in the area close to a minimizer. In order to find the Newton search direction for each iteration, a second order approximation to the point $x + z$ is made, where z is the search direction. The search direction is the direction along which the algorithm takes its next step. The algorithm takes this step with a certain step length, which is equal to one for pure Newton methods. The following Taylor approximation is the starting point for deriving the step length. The point x_i is viewed as a parameter since the interest lies in finding the step length z_i for this iteration.

$$h(x_i + z_i) = h(x_i) + Dh(x_i)z_i + \frac{1}{2}D^2h(x_i)(z_i, z_i) + o(\|z_i\|^2) \tag{2.7}$$

with z_i representing the Newton search direction for an iteration i. By defining $g(z_i) \cong h(x_i + z_i)$ one can write Equation (2.7) as in

$$g(z_i) := h(x_i) + Dh(x_i)z_i + \frac{1}{2}D^2h(x_i)(z_i, z_i). \tag{2.8}$$

When using the stationarity condition $Dg(z_i) = 0$ in (2.9), one can find the minimizer z_i. The stationarity condition states that when the gradient is zero in a certain point, this point is a local minimum. Therefore, it is necessary to find the direction z for which this condition holds.

$$Dh(x_i) + D^2h(x_i)z_i = 0 \tag{2.9}$$

Solving (2.9) for z_i gives the Newton search direction

$$z_i := -H^{-1}Dh(x_i). \tag{2.10}$$

With H denoting the Hessian, which is defined as $H := D^2h(x_i)$. The next point to be approximated can be calculated using the Newton search direction as shown in

$$x_{i+1} = x_i - H^{-1}Dh(x_i). \tag{2.11}$$

It may not be enough to only use the search direction to compute x_{i+1}. To ensure that the condition $h(x_{i+1}) < h(x)$ holds, it is necessary to modify the Newton method to include a varying step length, which determines how far the algorithm will move along the direction z_i for a certain iteration. For pure the pure Newton method, the step length is equal to one. When the step length one is maintained, it may not be sufficient for the Hessian to be positive definite, which is a requirement to ensure proper functioning of the Newton method. Modifying Equation (2.11) leads to

$$x_{i+1} = x_i - \alpha_i H^{-1}Dh(x). \tag{2.12}$$

The value of the step length α needs to be determined every iteration. The derivation of the variable step length is based on the method shown in [6]. The step length that is chosen has to satisfy the Armijo criterion shown in

$$h(x_i + \alpha z) \leq h(x_i) + c\alpha Dh(x_i)z. \tag{2.13}$$

For c a constant with values $c \in (0, 1)$. This condition is also called the sufficient decrease condition. It ensures that the step length causes a sufficient decrease in $h(x)$. A sufficient decrease is a decrease that is proportional to the step length α and the directional derivative $Dh(x_i)z$.

In order to find a step length to satisfy the Armijo condition, backtracking is used. This technique starts with a certain value for the step length. In the case of the Newton algorithm, one starts with step length $\alpha = 1$. One then checks if this step length satisfies Equation (2.13). If this is not the case and $h(x_i + \alpha z) > h(x_i) + c\alpha Dh(x_i)z$, α is multiplied by a factor p such that the new value for α is smaller than one. This process is repeated until a step length is found that satisfies the Armijo condition. An illustration of the backtracking algorithm is shown in Figure 2.4.

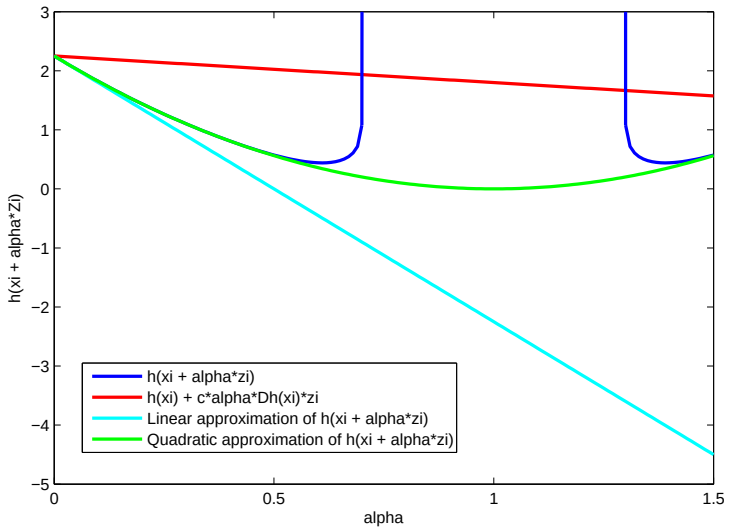


Figure 2.4: Finding the step length through backtracking

For the Newton step length $\alpha = 1$ the function value of $h(x_i + \alpha z_i)$ is infinite (see Figure 2.4) and therefore does not satisfy the Armijo condition. α is then multiplied by the constant p which is chosen to be 0.7. This value provides a large enough reduction of the step size each time the Armijo condition is not satisfied. As can be seen in the figure, the next value of α then becomes 0.7 as this point lies below the red line and therefore satisfies the Armijo condition. This means that for this iteration of the algorithm, the step length is chosen to be 0.7.

For each iteration, the algorithm will calculate a new search direction and step length in order to find the minimum. Each iteration a check is performed if the minimum has been reached. This minimum is considered reached when a certain point meets the termination criterion shown in Equation (2.14).

$$\|Dh(x_i)z_i\| \leq 10^{-6} \tag{2.14}$$

Previously was mentioned that the Newton algorithm does not guaranty descent when the Hessian H is no longer positive definite. Therefore, a check is performed to see if the Hessian is positive definite in every iteration. This is done by checking the eigenvalues of the matrix H for non-positive real values. If the Hessian is positive definite, the Newton search direction is accepted and the algorithm continues normally with the next step, which is to calculate the step length using the backtracking approach. However, if one or more of the real parts of the eigenvalues of H are non-positive, this means that the Hessian is negative definite or semi-positive definite. In this case, the Newton direction does not guarantee descent and a different method has to be chosen for this iteration [7]. The solution is to use the gradient search direction when the Hessian is not positive definite. This excludes the Hessian from the algorithm for this iteration and descent can be guaranteed. The point for the next iteration is then calculated by using Equation (2.6) instead of Equation (2.12).

The determination of the step length is still executed in the same way using the backtracking approach and the Armijo condition. However, a new stopping criterion is introduced.

$$\|x_{i+1} - x_i\| \leq 10^{-6} \tag{2.15}$$

As before mentioned, the algorithm is a relaxation of the Newton method and the interior point method. When the barrier parameter is made to approach zero, the minimum of the objective function $h(x_i)$ lies increasingly closer to the boundary as illustrated in Figure 2.5 for the 1D version of the objective function.

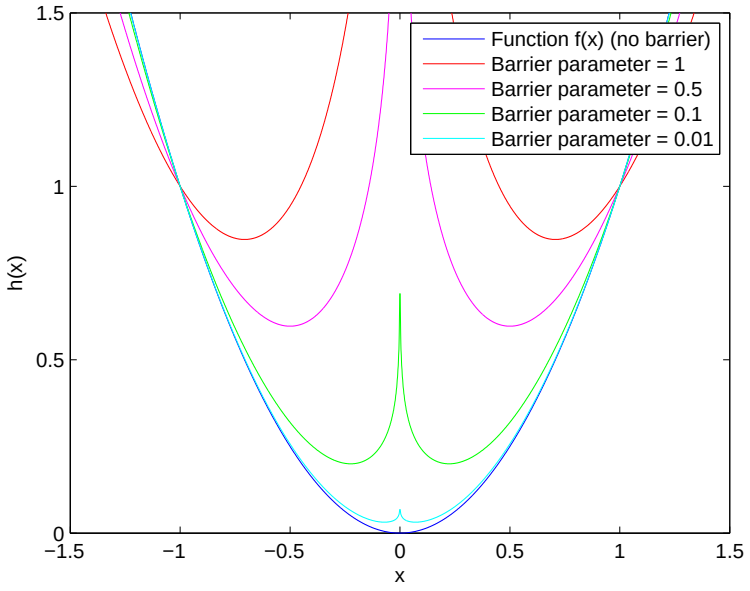


Figure 2.5: Illustration of the effect of a decreasing barrier parameter for the one-dimansional version of Equation (2.5)

As can be seen in the figure, the minimum of the objective function will lie closer to the unconstrained minimum when the barrier parameter approaches zero. For applications in motion planning where robots have to pass by objects, it is desirable that the robot is able to pass by those objects as closely as possible. Also in the case of a robot grasping an object, the robot should be able to come very close to the object. For these reasons ϵ should be able to become very small. In this report will be demonstrated that the algorithm converges with a sufficient accuracy for every value of ϵ.

2.4 Implementation

In the previous section, the algorithm has been defined. Here a short summary is given of how the algorithm works when implemented. In this description, the unconstrained minimizer x_f^* has been chosen as $x_f^* = [0, 0]^T$ and ϵ is a constant.

1. A starting value for x is chosen (denoted x_0).
2. For this value, the function values, the first derivative $Dh(x_0)$ and the Hessian H of $h(x_0)$ are calculated.
3. Next a check is performed to see if H has any non-positive eigenvalues in order to choose the search direction for this iteration. If the eigenvalues are positive, the Newton direction is chosen and otherwise the gradient descent direction is chosen.
4. When the direction has been determined, the step length is determined using the backtracking algorithm.
5. After the step length is determined, the point x_{i+1} can be calculated using either Equation (2.12) or Equation (2.6), dependent on which search direction was chosen.
6. Lastly, the algorithm checks if the current point is a minimizer by using either Equation (2.14) or Equation (2.15), dependent on which search direction was chosen.
7. If a minimizer has been found, the algorithm is terminated. If a minimizer was not found, the algorithm is repeated for x_{i+1}, starting at step 2.

2.5 Conclusion

This chapter started with synthesizing the grid values and making them twice continuously differentiable by using a cubic spline. After this, the optimization algorithm was shown. The algorithm is a relaxation of the Newton method and a log-barrier function. A varying stepsize has been implemented as well as the option to use the gradient descent in situations where the Hessian is not positive definite. In the next chapter, it is validated if the interpolated grid data lead to proper functioning of the algorithm.

3 Validation

In this chapter the algorithm is validated on a number of different aspects like the rate of convergence and different shapes. In the next chapter, the conclusion and recommendations are presented that follow from the validation in this chapter.

3.1 Validation experiments

The goal is to validate that the algorithm works properly for multiple situations, like different obstacle shapes and starting positions. The algorithm is considered to work properly when it converges for a termination criterion that is equal to the grid resolution. The algorithm should not have to converge more accurately than the resolution of the input data. The grid that has been used for this project is a square grid and has a value of 0.01 between each grid point, with an upper bound of 2.5 and a lower bound -2.5 in each direction. This means that the termination criterion should have a value of at least 10^{-2}. However, to show the robustness of the algorithm, the criterion has been chosen at 10^{-6} for the Newton termination criterion in Equation (2.14) and the termination criterion for the gradient descent in Equation (2.15).

The algorithm is validated for the ellipse as shown in Equation (2.1). A number of starting values were chosen to see if the algorithm could converge for all these positions, for a range of different ϵ. The range that is used is $\epsilon = 1, 0.5, 0.2, 0.01, 0.001$. For a few starting positions, the effect of continuation is shown. Continuation is used to reduce the number of iterations needed to converge. The concept of continuation is explained in Section 3.2.

Also a check of the convergence rate was done to see if the decrease is quadratic, as should be the case when the Newton algorithm is applied. To gain further insight in the rate of convergence, a circle was used as obstacle. The circle provided an interesting situation since it has global minima lying in a circle with a radius r_ϵ that is determined by the value of ϵ. It holds that $r_\epsilon > r$, $\forall \epsilon \in \mathbb{R}$, with r the radius of the obstacle. The circular distribution of the global minimizers should not lead to quadratic convergence as is explained in Section 3.2. The circular distribution of the global minima should provide insight into the rate of convergence.

The Cassini ovals are implemented using

$$u = (x_1^2 + x_2^2)^2 - (2c^2)(x_1^2 - x_2^2) - (a^4 - c^4); \tag{3.1}$$

Where a and c are the parameters that determine the shape of the ovals. The parameter values are chosen as $a = 1.005$ and $c = 1$. The validation that was done for this shape is the same as previously explained for the ellipse, with slightly different starting positions and the same values for ϵ. Also the continuation and the rate of convergence were analyzed.

Another validation for this shape was done by testing the algorithm for a different location of the unconstrained minimizer x_f^*. The validation that was done for this was purely to test the convergence for different starting values.

3.2 Validation on an elliptical shaped object

As described in Section 3.1, the first validation is done by checking the convergence for different initial values for x_0 for the range of ϵ shown in Section 3.1. The initial values x_0 are shown in Figure 3.1. The exact coordinates of the points plotted in the figure, can be found in Appendix A.

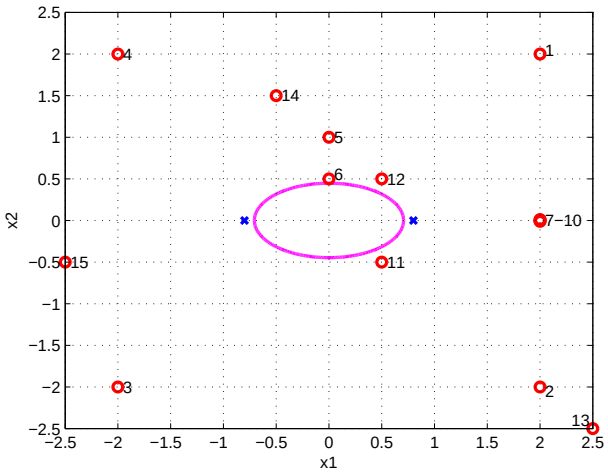


Figure 3.1: Initial values for the validation of the algorithm for the ellipse

There are two global minima located at both sides of the ellipse on the minor axis with equal distance to the major axis. The algorithm converges well for all of the observed ϵ and x_0. However, starting positions 7, 8, 9 and 10 provide an exception to this statement. These points lie close to the major axis of the ellipse. On the major axis close to the contour there exists a saddle point in the function $h(x)$ (see the blue crosses in Figure 3.1), which is encountered by the algorithm for the points 7-10. On this saddle point, the Hessian will have both positive and negative eigenvalues, which means that the algorithm switches to the gradient descent direction. Should x_0 be chosen directly on the major axis, this results in the derivative becoming sufficiently small to reach convergence on the saddle point. However, for points 7-10, x_0 lies a short distance away from the major axis. This distance is enough to prevent it from converging on the saddle point. Once the algorithm has passed the saddle point, the Newton method can be used again until the global minimum is reached. However, for small values of ϵ the algorithm takes a long time to converge for x_0 close to the major axis (more than 650 iterations for $\epsilon = 0.001$). This case presents the necessity to use continuation to reduce the number of iterations.

Continuation means starting with a large value for ϵ, finding a minimum and then continue searching for a minimum with a smaller value for ϵ. The ϵ_k are all ϵ used for the continuation process, with $k = 1, 2, ..., k \in \mathbb{N}$. The minimizer $x^*_{\epsilon_k}$ that is found when using ϵ_k is used as x_0 in $h(x_0)$ for ϵ_{k+1}. $x^*_{\epsilon_k}$ denotes the set of minimizers for ϵ_k.

In Figure 3.2, the minimizers $x^*_{\epsilon_k}$ for $\epsilon = 1, 0.5, 0.2, 0.01, 0.001$ have been plotted and their values are shown in Table 3.1. They approach the minimizers x^* for $\epsilon \to 0$, which lie on the contour of the ellipse in the points $x^* = [0, \pm\sqrt{0.2}]^T$. When applying continuation on the initial values close to the major axis, the number of iterations needed to converge, greatly decreases from more than 650 iterations to 29 iterations in total. The starting position of this continuation was point 8 in Figure 3.1.

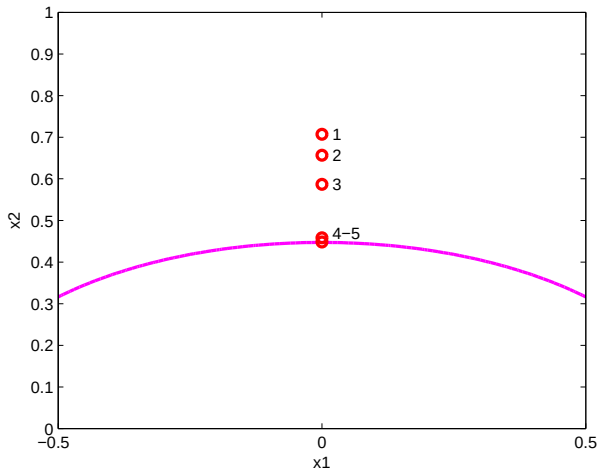


Figure 3.2: Continuation for $x_0 = [2, 0.01]^T$

Table 3.1: Minimizers for the continuation shown in Figure 3.2

k	ϵ_k	$x^*_{\epsilon_k}$
1	1	[-1.54e-6; 0.707]
2	0.5	[-4.23e-10; 0.657]
3	0.2	-1.42e-12; 0.587]
4	0.01	[-8.38e-13; 0.458]
5	0.001	[1.84e-14; 0.448]

Table 3.2 shows continuation for two other starting positions. In the first column, the number of iterations without using continuation are shown and in the second column the number of iterations it takes for the algorithm to converge with continuation are shown. This is for $\epsilon_3 = 0.2$ and $\epsilon_5 = 0.001$.

Table 3.2: Continuation versus constant ϵ for two starting positions x_0.

x_0	**Only** ϵ_2	**First** ϵ_3 **then** ϵ_5
[2.5;2.5]	12	13
[2;2]	11	13

It can be observed that the convergence is already quite fast for these starting positions with $\epsilon_5 = 0.001$, without continuation. Using continuation makes almost no difference. In fact, it seems to be slowing down the process when the x_0 lies closer to the boundary. A number of different combinations of ϵ_k have been tried and they all led to similar results. This gives an indication that when the convergence is already fast for a certain combination of ϵ and x_0, continuation does not necessarily make a large difference.

In order to conclude that the continuation indeed works properly for the situation at hand, it needs to be shown that $h(x^*_{\epsilon_k}) > h(x^*_{\epsilon_{k+1}})$. Figure 3.3 plots $h(x^*_{\epsilon_k})$ for a large range of ϵ.

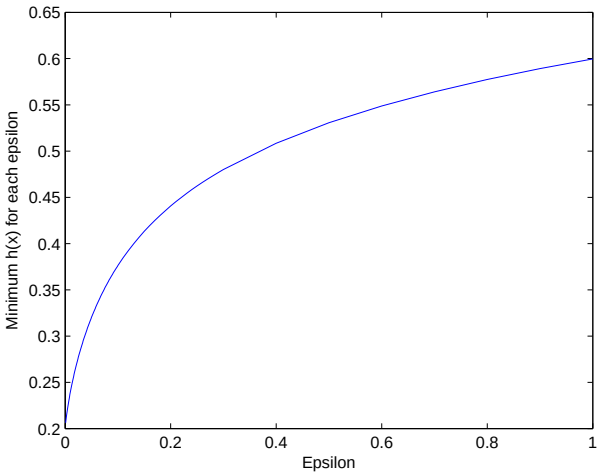
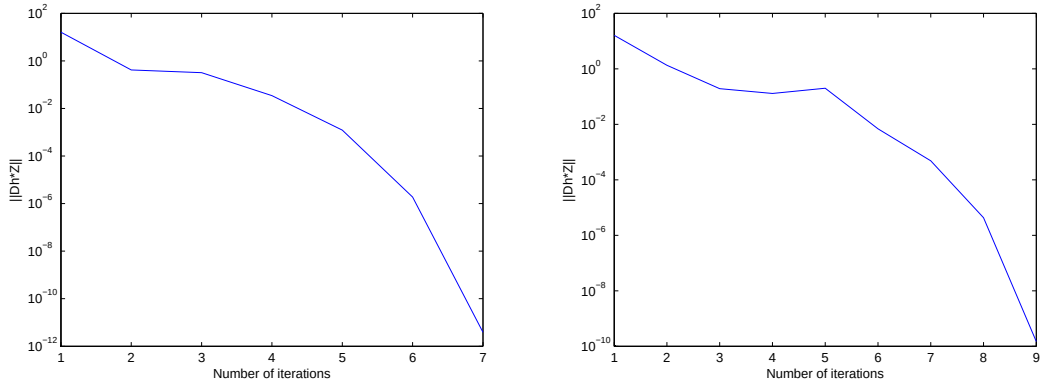


Figure 3.3: $h(x_\epsilon^*)$ approaching $h(x^*)$ as ϵ approaches zero.

When using the Newton algorithm, one should be able to observe quadratic decrease. This is validated by plotting $\|Dh(x_i)z_i\|$ against the number of iterations. The vertical axis is plotted on a log-scale in order to show the quadratic behavior.



(a) Decrease of h(x) for $\epsilon = 1$

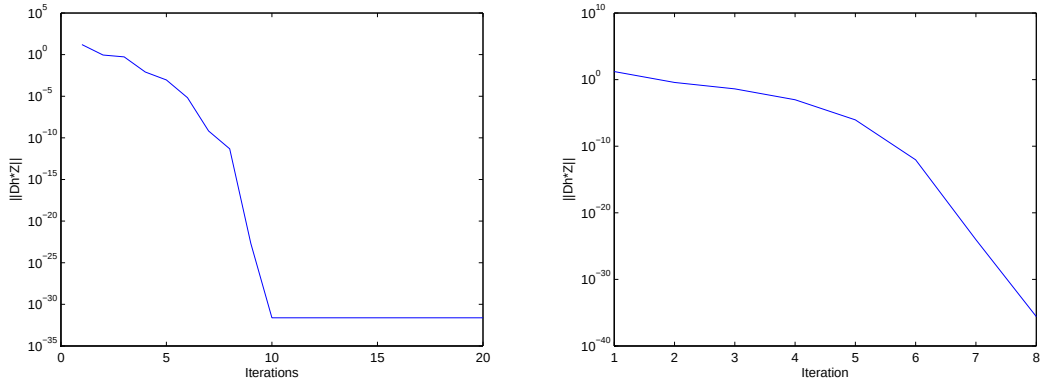
(b) Decrease of h(x) for $\epsilon = 0.01$

Figure 3.4: Decrease of h(x) for $x_0 = [2, 2]^T$

The plots have been created for a large value of ϵ ($\epsilon = 1$) and a small value for ϵ ($\epsilon = 0.01$). For $\epsilon = 1$, the algorithm first takes a large step to the area near the obstacle. For the second iteration, quadratic decrease can clearly be observed. When $\epsilon = 0.01$, it takes about two steps to approach the area of the minimum near the obstacle. The chosen starting positions will cause the algorithm to approach the saddle point mentioned before, which causes the behavior observed between iterations 3 and 5 in Figure 3.4b.

In Chapter 3.1 was mentioned that the circle could provide an interesting validation for the rate of decrease because of the location of the global minima. The Newton method ensures local quadratic convergence when the second order sufficient condition for optimality is met at the minimizer. This condition states that the Hessian should be positive definite at the minimizer. This is the case when the minimizer is a strict minimizer. In other words, the minimizer lies in a location where there are no other minimizers nearby [6]. For the circle, this is not the case because the minimizers lie adjacent to one another on a circle with radius r_ϵ as explained in 3.1. Therefore, one should not be able to observe quadratic convergence. In order to illustrate the difference between the convergence of the circle and the ellipse, the convergence of the two shapes was plotted for the same x_0 and ϵ. In order to best show the difference in convergence rate, the termination criterion was changed to 10^{-32} for this validation. In Figure 3.5, the decrease for the circle and for the

ellipse are plotted.



(a) Convergence for the circle

(b) Convergence for the ellipse

Figure 3.5: Decrease of h(x) for $x_0 = [2, 2]^T$

In order for the circle to converge for this termination criterion, more than 500 iterations were needed. For illustration purposes, only the first 20 iterations are plotted in Figure 3.5a. For the ellipse only 8 iterations were needed before the algorithm found the minimizer. It can be concluded from this observation and from the shape of the graphs that there indeed is no quadratic decrease for the circle, which corresponds to the second order sufficient condition for optimality.

3.3 Validation on a Cassini oval shaped object

In the same way as for the ellipe in Section 3.2, the convergence was checked for different initial values for x_0 as shown in Figure 3.6. Appendix A contains the exact coordinates of the points shown in the figure.

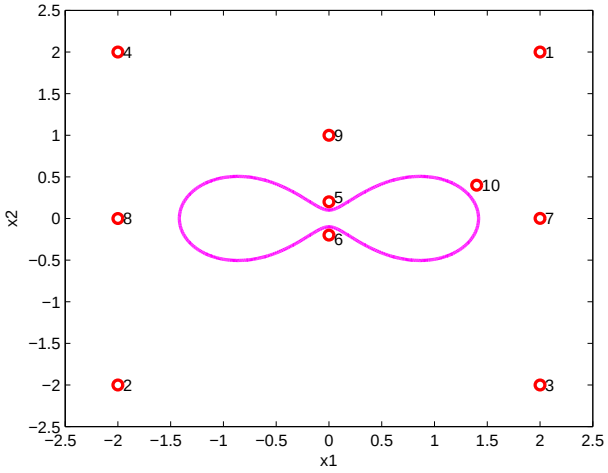


Figure 3.6: The starting values x_0 of the algorithm validation of convergence

These starting positions were tested for $\epsilon = 1, 0.5, 0.2, 0.01, 0.001$, to see if they converge to the minimizers which lie outside the contour on the minor axis. The algorithm converges each time, but for this shape there are again a saddle points present on the major axis of the oval on either side of the obstacle. Passing this point again takes a lot of iterations for smaller ϵ (more than 1000 iterations). Continuation reduces this to 25 iterations. The continuation steps are is shown in Figure 3.7 and the minimizers are listed in Table 3.3.

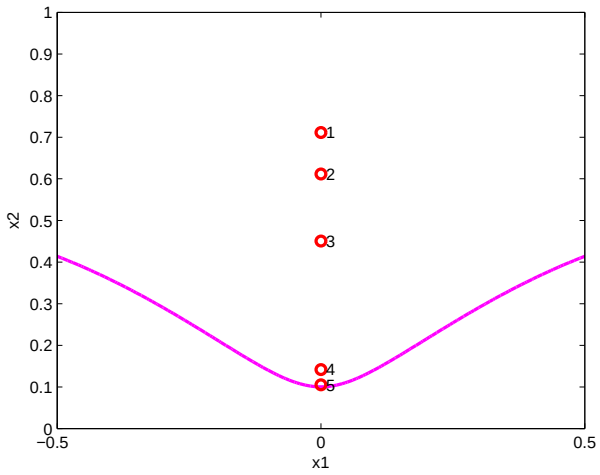


Figure 3.7: Continuation for $x_0 = [2, 0.001]^T$

Table 3.3: Minimizers for the continuation shown in Figure 3.7

k	ϵ_k	$x^*_{\epsilon_k}$
1	1	[3.72e-5; 0.711]
2	0.5	[-1.87e-7; 0.611]
3	0.2	1.54e-9; 0.451]
4	0.01	[-4.82e-13; 0.142]
5	0.001	[6.88e-16; 0.105]

Continuation was also tested for two other starting positions $x_0 = [2.5, 2.5]^T$ and $x_0 = [2, 2]^T$. The results are shown in Table 3.4. The chosen values for ϵ for the continuation are $\epsilon_1 = 0.2$ and $\epsilon = 0.001$.

Table 3.4: Continuation versus constant ϵ for two starting positions x_0.

x_0	**Only** ϵ_2	**First** ϵ_1 **then** ϵ_2
[2.5;2.5]	12	12
[2;2]	9	12

From this, the same conclusion can be drawn as for the ellipse. The continuation improves the performance most if the number of iterations when using a small constant value for ϵ is large. This validates that the conclusion holds both shapes and might indicate robustness.

In Figure 3.8 it is validated that $h(x^*_\epsilon)$ decreases and approaches $h(x^*)$ as $\epsilon \to 0$. This result corresponds to Figure 3.3, which again indicates robustness.

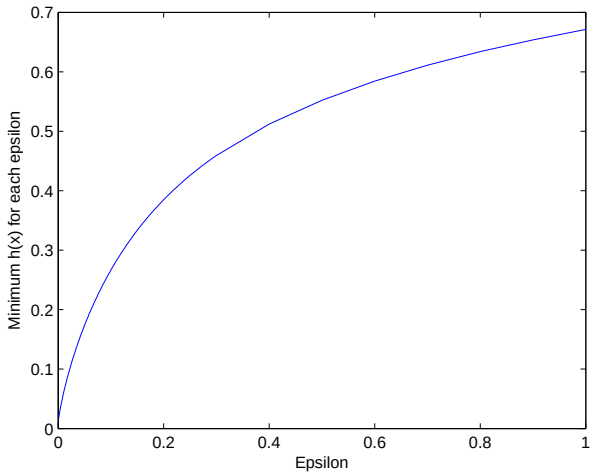
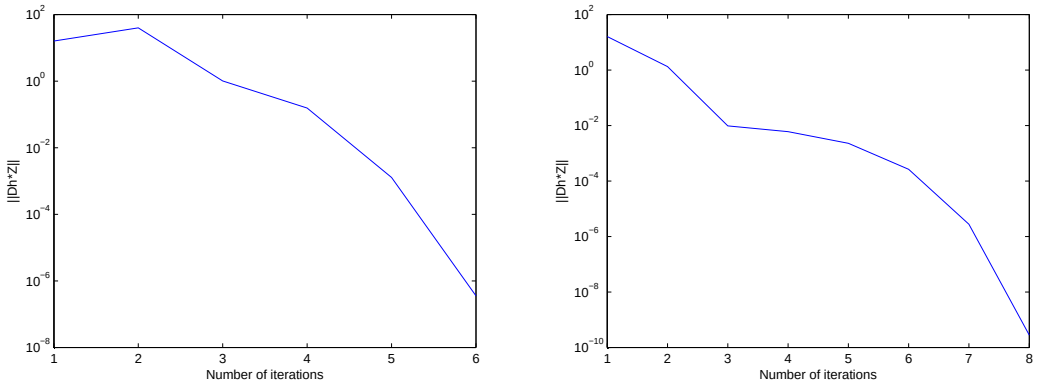


Figure 3.8: $h(x_\epsilon^*)$ approaching $h(x^*)$ as ϵ approaches zero.

Also for this Cassini oval can be shown that there is quadratic decrease.



(a) Quadratic decrease for $\epsilon = 1$.

(b) Quadratic decrease for $\epsilon = 0.01$.

Figure 3.9: Quadratic decrease for $x_0 = [2, 2]^T$, for $\epsilon = 1, 0.01$.

One last validation can be done to see how the algorithm converges if the unconstrained minimizer, is placed as shown by the blue cross in Figure 3.10. This Figure also shows the starting positions for which the convergence was checked for the range of ϵ that was also used for the ellipse and Cassini ovals.

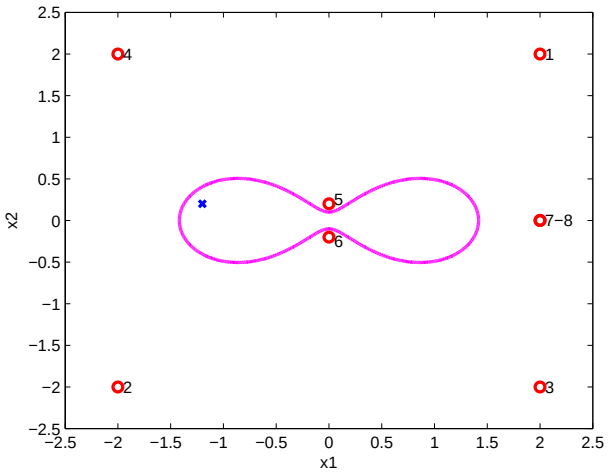


Figure 3.10: Initial values for x_0 for the unconstrained minimum $[-1.2, 0.2]^T$

This point does not lie on the major or minor axis of the Cassini oval and has no saddle points. This is confirmed by the fact that convergence is quick for starting positions near the major axis. When the unconstrained minimum was located in the middle of the oval, this resulted in over 1000 iterations. With the unconstrained minimum placed at $x = [-1.2, 0.2]^T$, starting positions 7 and 8 only need 12 iterations.

3.4 Conclusion

In this chapter, it was validated that the algorithm converges as expected for a Newton-like algorithm for three different shapes. The algorithm is able to converge for small termination criteria. Also, the rate of convergence was as expected quadratic for the ellipse and the Cassini oval and non-quadratic for the circle. For the Cassini oval it was also validated that the algorithm was able to converge for a different position of the unconstrained minimizer. The next chapter gives conclusions about the findings during this project and recommendations for future research.

4 Conclusions and recommendations

4.1 Conclusions

The goal of this project was to investigate whether the grid data containing a volumetric function could be made twice continuously differentiable such that it could be employed to Newton-like optimization algorithms. To this end, the grid data was made twice continuously differentiable by interpolation using a planar cubic spline. Next, an algorithm was created for constrained optimization problems using a relaxation of the Newton-method and a barrier function. The input for the log-barrier function is the interpolant that resulted from the interpolation of the grid data. By using this logarithmic-barrier function as part of the objective function as input for the Newton-method, the grid data was required to be twice continuously differentiable. The validation in Chapter 3 shows that the algorithm is capable of converging for different implicit functions embedded in the grid data. The algorithm was able to converge for different scenarios. It was shown that there was local quadratic decrease for every shape except for the circle as expected from the theory of the second order sufficient condition for optimality. The effect of continuation was validated for the ellipse and the Cassini oval where the unconstrained minimizers were located at the center of the shape. The continuation was shown to be especially useful in situations where many iterations were required for the algorithm to converge and less useful where the algorithm already converged within only 12 iterations.

These results show that the algorithm with the input from the interpolated grid data, behaves as expected. From this can be concluded that the grid data created with volumetric methods was made twice continuously differentiable to be able to be implemented in the Newton-like algorithm presented in this report.

4.2 Recommendations

This project has focused solely on the 2D case. However, when this project is extended to include input from models that portray the surroundings of a robot or camera data containing the 3D voxel grid, the algorithm needs to be shown to work for a 3D case. In order to be able to use the findings in this report for practical applications in motion planning in the future, the extension to 3D environments is crucial.

Throughout the project the same grid has been used for all the computations. However, the grid size has an influence on the representation of the shapes. If a shape has sharp edges, they may not be represented accurately on a coarse grid. This could influence the reliability of the results and therefore, the influence of the grid size on the accuracy of the approach should be investigated further.

So far, this project has revolved around minimizing the distance to a certain point from a single starting position x_0. However, an application that this algorithm could be used for is a robot arm with a gripper. For this application one point is not sufficient to represent the gripper arm approaching an object. A suggestion is to use wireframe objects and apply this to the algorithm to see if the algorithm is usable for this type of application.

References

[1] S S Antman, J E Marsden, and L Sirovich. *Level Set Methods and Dynamic Implicit Surfaces*, volume 153. Springer, 2003.

[2] Richard H Bartels, John C Beatty, and Brian A Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modelling*. Morgan Kaufmann Publishers, Inc, 1987.

[3] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96*, pages 303–312, 1996.

[4] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera *. *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011.

[5] Carsten Maple. Geometric Desing and Space Planning Using the Marching Squares and Marching Cube Algorithms Carsten Maple Departement of Computing and Information Systems. *International Conference on Geometric Modeling and Graphics*, pages 90–95, 2003.

[6] Jorge Nocedal and Stephen J Wright. *Numerical Optimization*. Springer, second edition, 2006.

[7] Panos Y. Papalambros and Douglas J. Wilde. *Principles of Optimal Design: Modeling and Computation*. Cambridge University Press, second edition, 2000.

[8] David H. von Seggern. *CRC Standard Curves and Surfaces with Mathematica*. CRC Press, Boca Raton, 2016.

[9] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. CHOMP : Covariant Hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013.

A Experiment details

In this appendix, the initial values for the validation of the ellipse and the Cassini ovals are shown. Table A.1 shows the initial values for the ellipse, Table A.2 shows the initial values for the Cassini oval and Table A.3 shows the initial values for the Cassini oval for the replaced unconstrained minimizer.

Table A.1: Initial values for the validation of the algorithm for the ellipse

	x_0
1	[2;2]
2	[2;-2]
3	[-2;-2]
4	[-2;2]
5	[0;1]
6	[0;0.5]
7	[2;-0.01]
8	[2;0.01]
9	[2;-0.001]
10	[2;0.001]
11	[0.5;-0.5]
12	[0.5;0.5]
13	[2.5;-2.5]
14	[-0.5;1.5]
15	[-2.5;-0.5]

Table A.2: Initial values for the validation of the algorithm for the Cassini oval

	x_0
1	[2;2]
2	[-2;-2]
3	[2;-2]
4	[-2;2]
5	[0;0.2]
6	[0;-0.2]
7	[2;0.001]
8	[-2;-0.001]
9	[0;1]
10	[1.4;0.4]

Table A.3: Initial values for the validation of the algorithm for the Cassini oval with replaced unconstrained minimizer

	x_0
1	[2;2]
2	[-2;-2]
3	[2;-2]
4	[-2;2]
5	[0;0.2]
6	[0;-0.2]
7	[2;0.001]
8	[2;-0.001]