A HOMOTOPY METHOD FOR MOTION PLANNING

BY

YUYING LI

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science in Electrical and Computer Engineering
in the College of Engineering at the
University of Illinois at Urbana-Champaign, May 2018

Advisor:

Professor Mohamed-Ali Belabbas
ABSTRACT

The field of robotics has developed at a promising speed these years, and its potential usage is tremendous in both traditional industry and frontier research. Motion planning, as an important branch of robotics, is the process of completing tasks and avoiding obstacles while achieving optimization at the same time. Even though many ad hoc algorithms have been proposed to solve such problems, most of them will deal constraints and optimization separately. As a result, there is an urgent need for an algorithm to handle them together.

With the aim to solve this problem, this paper introduces a new method with the help of homotopy, and such a method and its application for motion planning is examined comprehensively. This method will assemble all holonomic, non-holonomic, and obstacle constraints into a metric. Furthermore, applying partial differential techniques on this metric allows continuously deforming an arbitrary path into another one in the configuration space. A shortest path will be found that satisfies all given constraints such as obstacles and implementation requirements. The corresponding mathematical and control background will be reviewed. Then this thesis will present the details of the homotopy method and how to implement it in reality. Finally, two models with different degrees of freedom will be examined to look at the feasibility of such a method.

Keywords: motion planning; non-holonomic system;
To my parents, for their love and support.
# TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION ........................................... 1

CHAPTER 2 FUNCTIONAL AND EULER-LAGRANGE EQUATIONS ........................................... 3
   2.1 Euler Langrange ........................................... 3
   2.2 Introduction of Functional ..................................... 6
   2.3 Introduction of Euler-Lagrange Equation ................................. 9
   2.4 Euler-Lagrange with Constraints .................................... 12

CHAPTER 3 CONTROL MODEL ........................................... 14
   3.1 State-Space Control Model ....................................... 14
   3.2 Admissible Control ........................................... 17

CHAPTER 4 HOMOTOPY METHOD FOR MOTION PLANNING 20
   4.1 Constraints and Metric G ....................................... 20
   4.2 Homotopy Method ........................................... 24

CHAPTER 5 IMPLEMENTATION AND RESULTS ........................................... 26

REFERENCES ........................................... 29
CHAPTER 1
INTRODUCTION

The development of the field of robotics is going faster than our imagination. Ten years ago, it was hard to believe that a robot can simulate human locomotion to walk, jump, and do backflips in a smooth way, where the latter one can’t even be done by some people. Furthermore, the flourishing of many related fields such as computer vision and machine learning helps open the ”eyes” and the learning ability of robots, which expands its potential in both the research and application realms. In recent years, applications of autonomous machines such as driverless car technology by automotive companies and drones further broaden the definition of robotics. It is obvious that robots will play more important roles in the future.

Literature Review  Motion planning, as an crucial branch of robotics, has attracted lots of attention. As an unavoidable issue of robotics application in reality, motion planning tries to solve the problem that allows robots to move along a feasible path in both the real world and the configuration space while pursuing optimization. Much research has been conducted and a lot of algorithms are proposed trying to solve such problems. Related books [1, 2, 3] give thorough tours of various planning algorithms in details. Furthermore, recent research paper also proposed many ad hoc solutions for this problem. One approach is finding a feasible path in the configuration space [3] using interval analysis [4, 5]. There are also some techniques such as bug algorithms [6] and cell decompositions. Yet, most of them isolate optimization and constraints such that they treat either one as a special case when put majority efforts trying to solve another one. In particular, non-holonomic systems with obstacle avoidance needs [7, 8] is always a far more difficult problem than we thought. As a result, this thesis will present a novel method to solve such non-linear systems with tolerance of both optimization and constraints as part of the method’s nature.
This method utilizes the concept of homotopy drawn from topology. For two continuous functions from a common topological space, one function can deform into another one and such deformation is called homotopy between these two functions. The same concept can be applied to robots in configuration space, which have starting and final states. We are able to obtain a desired path by solving partial differential equations with Euler-Lagrange constraints based on a randomly generated holonomic and non-holonomic constraint-free path.

Specifically, in Chapter 2, math tools will be reviewed. At the beginning of this chapter, we will recall some calculus about solving constraint problem for a function. Then a crucial concept in calculus of variation: functional will be present. Finally, a thorough derivation to find optimization (under constraints) of functional and its application is given.

Chapter 3 then gives a brief summary of the state-space control model. Then, the definition of admissible control will be displayed. Adequate examples of finding such control are also used to illustrate the concept properly.

The formal subject of homotopy method will be extended in Chapter 4. This chapter starts with a clarify for the relationship among control, optimization and motion planning. Then starting from introducing various constraints that will be met during motion planning designing, a metric is offered to involve all given constraints. Implementation will show the feasibility and easiness to compute such metric. Finally, both the metric and the above-mentioned concepts such as Euler-Lagrange constraints and admissible control are combined to form the new method.

In the last chapter, implementations and the results of two models are given. The first model is a unicycle with three degrees of freedoms. We show how it can incorporate the proposed method to adapt to different task environments. A more complex robot legs with details are given as a second cases to demonstrate. Discussion of result for both implementations will show the achievable and ability for future usage of such method.
CHAPTER 2

FUNCTIONAL AND EULER-LAGRANGE EQUATIONS

When designing the path for robots, for example, finding a path for a robot to reach a bottle of water, it is trivial to achieve the goal if no obstacles exist. All the robot needs to do is going a straight line to touch the bottle. However, if some obstacles are placed along the way, it becomes a harder problem. Furthermore, other than satisfying given constraints, optimization on time or energy is another objective because of task efficiency purpose. The most straightforward way to find such optimization is using calculus and differential equation. As a result, before going in details into the analysis of motion planning problem, some background mathematical knowledges need be reviewed and expanded to provide necessary tools in subsequent work. In the following sections, the Lagrange multiplier will be first reviewed which helps finding satisfied optimized solution for given constraint(s) of a function. Then the definition of functional, which in short can be defined as ”function of functions” is introduced. Functional changes input variables from real values to function and it will be helpful when in latter paper, path becomes the variables. Finally, Euler-Lagrange equations arise from calculus of variations are addressed to solve the optimization problem about functionals.

2.1 Euler Multiplier

When problems like finding extrema are waiting for a solution, the intuition tells us to think back to calculus [9]. For any point of a function \( f : \mathbb{R} \rightarrow \mathbb{R} \), the derivative at any point \( x : x \in \mathbb{R} \) is defined as:

\[
\nabla f(x) = \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}
\]

(2.1)

where h is a small amount of increment and we assume it is always positive for simplicity. The symbol \( \nabla f(x) \) represents derivative of function \( f(x) \) with
respect to $x$.

The curve of function $f(x)$ is on a two-dimensional Euclidean plane, and $\nabla f(x)$ is only defined on the segment of curves that are continuously differentiable. Based on (2.1), if $f(x + h)$ is a value greater than $f(x)$, the positive value resulting the derivative $\nabla f(x) > 0$. Similarly, $\nabla f(x) < 0$ if $f(x + h)$ is smaller than $f(x)$.

Therefore, the value of derivative for a given function $\nabla f(x)$ always changes sign when it encounter local extrema, either minimum or maximum. For example, say for $f(x_0)$ is a local minimum and it equals some constant $c \in \mathbb{R}$: $f(x_0) = c$. Defining an infinitesimal but non-zero increment $\delta$, and claim that this increment will not affect the function value, shown as (2.2). It is not hard to see from figure 2.1 that if $\delta < 0$, we have a negative derivative because the function is always decreasing from the left side. Similarly, the derivative is positive if $\delta > 0$ because of increasing function value. Therefore, the local minimum is a turning point that derivative goes from negative to positive. So the derivative value at this turning point is 0.

![Figure 2.1: Function $f(x)$ with small change $\delta$ around its minimum $x_0$](image)

$$\lim_{\delta \to 0} f(x_0 + \delta) = c$$

The Taylor expansion of $f(x_0)$ (2.3) can further be applied to verify the
value of its derivative. Combining (2.2) and (2.3), we found \( \nabla f \cdot \delta = 0 \). Since \( \delta \neq 0 \) based on former definition, the only solution meets the requirement is \( \nabla f = 0 \), which consistent with previous conclusion.

\[
f(x_0 + \delta) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!} \approx f(x_0) + \nabla f \cdot \delta = c
\]

As a result, setting the derivative of function equals to zero, we can find local extrema and denote it as \( r \). This method is called first derivative test, shown as (2.4). It is also important to note that the result obtained from (2.4) not only contains local minimum and maximum, but also saddle points. To distinguish differences among them, we need further tests. However, because saddle points are irrelevant with the topic presented in this paper, we will omit relevant discussion here.

\[
r = \{ x | \nabla f(x) = 0 \}
\]

Another question arises after successfully finding local extrema is how to find extrema when imposing constraints. For example, if there exists a function \( \{ f(x,y) : X \to \mathbb{R} | X \in \mathbb{R}^2 \} \) and another constraint function: \( \{ g(x,y) : X \to \mathbb{R} | X \in \mathbb{R}^2 \} \), where \( X \) is set of all tuple \((x,y)\) such that \( x,y \in \mathbb{R} \). Assuming both \( f \) and \( g \) have continuous first derivative, what is the method to find such solution. Once again, we can visualize both \( f(x,y) \) and \( g(x,y) \) as function curves, shown as Fig.2.2, where blue dashed lines are curves of \( f(x,y) = d_i \) for different values \( i \in \{1,2,3,\ldots\} \) and red solid line represents constraint \( g(x,y) = c \). The arrows around each level curve are the normals of corresponding level curves, where the direction of each arrow can be calculated by taking gradients of the original function, shown in equation (2.5) and (2.6).

\[
\nabla_{x,y} f = \left( \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right) \quad (2.5)
\]

\[
\nabla_{x,y} g = \left( \frac{\partial g}{\partial x}, \frac{\partial g}{\partial y} \right) \quad (2.6)
\]

Furthermore, since the goal is finding a point \((x_0,y_0)\) that satisfies constraint and optimize original function, we can follow the contour of constraint \( g \) to find points where function \( f \) does not change. Recall from previous results, these points are candidates of extremas for \( f \). Since at these points,
two curves coincide, the normals of both should also be parallel to each other.

In order to find the exact point \((x_0, y_0)\), the Lagrange multiplier \(\lambda\) is introduced. A Lagrange multiplier is a constant used to express the relationship of normals between \(f(x, y)\) and \(g(x, y)\). Since even though we know the normals are parallel, there is no guarantee about their magnitudes and directions. However, one thing assured is there exists a scalar linear relationship between the two normals in order to maintain their parallel properties. Therefore, \(\lambda\) can be either positive or negative as long as it satisfies following equations:

\[
\nabla f(x, y) = \lambda \nabla g(x, y) \tag{2.7}
\]

\[
\nabla f(x, y) - \lambda \nabla g(x, y) = 0 \tag{2.8}
\]

\[\]

2.2 Introduction of Functional

In last chapter, we found a way to find extrema of a given function under constraint using a Lagrange multiplier. One more step can be taken from here: we will introduce functional. Functional [10] is a core concept in calculus of variations, which is a field of mathematical analysis that deals with derivative and integral of functions. An analogy between function and functional is helpful to build our understandings. Imagining function is a ‘black box’ whose output is in real or complex domains and the input is another set of real or complex space. The ‘black box’ maps the input to another value: \(f : \mathbb{R}/\mathbb{C} \to \mathbb{R}/\mathbb{C}\). Functional is a similar ‘black box’ except its argument is a function. Functional takes functions as inputs to some higher-level functions.
It maps a vector space into real or complex values: $g : V \to \mathbb{R}/\mathbb{C}$, where the vector space $V$ is a space of functions. A formal definition of functional based on Kolmogorov [11] is given: a functional is a mapping $f$ of an arbitrary set $X$ into the set $\mathbb{R}$ of real numbers or the set $\mathbb{C}$.

![Figure 2.3: Function and Functional Black Box Model](image)

(a) Function (b) Functional

However, most of the time when dealing with the calculus problem about functional, what we are interested is finding maximum and minimum functions. It is not surprising to call up the techniques of derivatives and the Euler multiplier described in Section(2.1). Finding the extrema of a functional should be akin to find extreme point of a function. But the question is, how to solve this differential problem with respect to a function.

![Figure 2.4: Points A and B with $dx$ and $dy$ increments](image)

Let’s use an example to figure out the proposed problem. Suppose on a two-dimensional plane, there are two points: $A = (x_1, y_1)$ and $B = (x_2, y_2)$, shown in Fig. 2.4. There are some relations between $x_1$ and $y_1$ as well as $x_2$ and $y_2$ such that they’re both one-to-one mappings. Therefore, $y_1$ can be
represented as \( y_1 = f(x_1) \) and same can be applied for \( y_2 \). Then, the arisen question is finding a path \( I \) between \( A \) and \( B \) such that \( I \) is minimized. In order to exhibit the problem, we denote \( dS \) as a infinitesimal increment goes from \( A \) to \( B \) based on \( dx \) and \( dy \), where \( dS \) equals:

\[
\begin{align*}
  dS &= \sqrt{dx^2 + dy^2} \\
  &\quad \quad \quad \quad \quad \quad (2.9)
\end{align*}
\]

Taking \( dx \) out of the square root sign, \( dS \) can be rewritten as:

\[
\begin{align*}
  dS &= \sqrt{\frac{dx^2}{dx^2} + \frac{dy^2}{dx^2} dx} = \sqrt{1 + \left( \frac{dy}{dx} \right)^2 dx} \\
  &\quad \quad \quad \quad \quad \quad (2.10)
\end{align*}
\]

Since this path connects points \( A \) and \( B \), and \( dS \) is the infinitesimal change on the path, we can formulize \( I \) based on \( dS \) as:

\[
I(f(x)) = \int_A^B dS
\]

Substituting \( dS \) with the one from (2.10), we will get:

\[
\begin{align*}
  I(f(x)) &= \int_A^B \sqrt{1 + \left( \frac{dy}{dx} \right)^2 dx} \\
  &= \int_{x_1}^{x_2} \sqrt{1 + \left( \frac{dy}{dx} \right)^2 dx} \\
  &\quad \quad \quad \quad \quad \quad (2.12)
\end{align*}
\]

The step goes from (2.12) to (2.13) is due to the fact that \( dx \) is the one variable influences the integral of \( I \). Furthermore, observing these two equations, the problem of finding minimize path \( I(f(x)) \), which is a functional, reduces to find a function \( y = f(x) \) between points \( x_1 \) and \( x_2 \) such that the integral \( I(f(x)) \) is minimized.

Suppose there is a particle moving along the path \( I \). We can define \( v(x, y) \) as the velocity of this particle and the question becomes more interesting as finding solutions to minimize the time cost: \( T \). From relationship between time and velocity, we can write the amount of time needed to pass certain distance \( dS \):

\[
\begin{align*}
  dt &= \frac{dS}{v(x, y)} \\
  &\quad \quad \quad \quad \quad \quad (2.14)
\end{align*}
\]
Therefore, this problem becomes:

\[ T = \int_{x_1}^{x_2} \sqrt{1 + \left(\frac{dy}{dx}\right)^2} \frac{v(x, y)}{v(x, y)} \, dx \]  

(2.15)

Another example is dealing with more complex functions. Supposing there is a functional \( J \) with a function \( \bar{y}(x) \) who has input variable \( x \). \( \bar{y}(x) \) is twice continuously differentiable on interval \([a, b]\) where \([a, b] \subset \mathbb{R}\). Furthermore, there is another higher order function \( F(x, \bar{y}, \bar{y}_x) \), which is also twice continuously differential with respect to \( x, \bar{y} \) and \( \bar{y}_x \). The relation between \( J(\bar{y}) \) and \( F(x, \bar{y}, \bar{y}_x) \) is shown in (2.16), where \( \bar{y}_x = \frac{d\bar{y}}{dx} \). We want to discover a function \( \bar{y}_{min} \) such that \( j(\bar{y}) \) is minimized.

\[ \bar{y} = \begin{cases} 
   y_1 \\
   y_2 \\
   \vdots \\
   y_n 
\end{cases} \text{ which extremizes } J(\bar{y}) = \int_a^b F(x, \bar{y}, \bar{y}_x) \, dx \]  

(2.16)

In general, formulizing problem properly is the first step we need do to solve functionals. Most of the case, such problems share a common pattern similar to the integral (2.17). Once a nice and well-organized integral expression is obtained, following procedures are straightforward. For clarity, in (2.17), \( J \) is a functional while both \( f \) and \( F \) are functions with different arguments.

\[ J[f] = \int_{x_1}^{x_2} F(x, y, \frac{dy}{dx}) \, dx \]  

(2.17)

2.3 Introduction of the Euler-Lagrange Equations

The concept of functional was proposed in the previous section and the first step to formalize such problems is also provided. A remaining dilemma is how to actually find the solution \( y = f(x) \) such that \( J[f] \) in (2.17) is at its extrema. Referring to Section(2.1), we know setting the derivatives of a function equal to zero can find extrema for a function. Similarly method applies here. Using the notation in (2.17), supposing \( J(y(x_0)) \) is the local minimum of functional \( J \). Let’s have a small function increment \( h(x) \) such
that \( h(x) \) is non-zero only around \( x_0 \). Furthermore, let \( \Delta \sigma \) be the area between \( h(x) \) and the curve connecting points \( A \) and \( B \), which is \( y(x) \), shown as Fig. 2.5. Then a duality of derivative to (2.1) will have:

![Figure 2.5: Minimum function \( y(x_0) \) and incremental function \( h(x) \)](image)

\[
\nabla J(y(x)) = \lim_{\Delta \sigma \to 0} \frac{J(y(x_0) + h(x)) - J(y(x_0))}{\Delta \sigma}
\]

(2.18)

There is an arbitrary twice continuously differentiable function \( \eta(x,y) \) satisfies that \( \eta(x_1,y_1) = \eta(x_2,y_2) = 0 \). Then for \( \varepsilon \ll 1 \), there exists:

\[
J(y) \leq J(y + \varepsilon \eta)
\]

(2.19)

Rewrite functional \( J(y + \varepsilon \eta) \) as a function of \( \varepsilon \) with some changes in notation, we will have:

\[
\Phi(\varepsilon) = J(y + \varepsilon \eta)
\]

(2.20)

Because \( y = y(x_0) \) is the minimum function, then function \( \Phi(\varepsilon) \) has its minimum when \( \varepsilon = 0 \), which can be represented in derivative form: (2.21).

\[
\nabla \Phi(0) = \frac{d \Phi}{d \varepsilon} |_{\varepsilon=0} = \frac{d}{d \varepsilon} J(y + \varepsilon \eta)
\]

(2.21)

Combining (2.17) into (2.21) and substituting \( y \) by \( y^\varepsilon = y + \varepsilon \eta \), (2.21) becomes:

\[
\nabla \Phi(0) = \frac{d}{d \varepsilon} J(y + \varepsilon \eta) = \frac{d}{d \varepsilon} \int_{x_1}^{x_2} F(x, y^\varepsilon, y_x^\varepsilon) dx
\]

(2.22)
Taking the partial derivative of \( F(x, y^\epsilon, y_x^\epsilon) \) with respect to \( \epsilon \), we have:

\[
\int_{x_1}^{x_2} \frac{\partial}{\partial \epsilon} F(x, y^\epsilon, y_x^\epsilon) dx = \int_{x_1}^{x_2} \left( \frac{\partial F}{\partial y^\epsilon} \frac{\partial y^\epsilon}{\partial \epsilon} + \frac{\partial F}{\partial y_x^\epsilon} \frac{\partial y_x^\epsilon}{\partial \epsilon} \right) dx
\] (2.23)

Since \( y^\epsilon = y + \epsilon \eta \), we have \( \frac{\partial y^\epsilon}{\partial \epsilon} = \eta \) by taking derivative of \( y^\epsilon \) with respect to \( \epsilon \). Moreover, we can apply integral by parts to the second term of this partial derivative equation (2.23):

\[
\nabla \Phi(0) = \int_{x_1}^{x_2} \frac{\partial F}{\partial y} \eta dx + \frac{\partial F}{\partial y_x} \eta \bigg|_{x_1}^{x_2} - \int_{x_1}^{x_2} \eta \frac{d}{dx} \frac{\partial F}{\partial y_x} dx
\] (2.24)

We know the second term of (2.25) is 0 since \( \eta(x, y) \) is a function that vanish at points \( A \) and \( B \): \( \eta(x_1, y_1) = \eta(x_2, y_2) = 0 \) by early claims. Additionally, since \( \epsilon = 0 \), \( y^\epsilon = y + \epsilon \eta \) is as same as \( y^\epsilon = y \). Therefore, after rearranging (2.25), it becomes:

\[
\nabla \Phi(0) = \int_{x_1}^{x_2} \left( \frac{\partial F}{\partial y} - \frac{d}{dx} \frac{\partial F}{\partial y_x} \right) \eta dx = 0
\] (2.26)

The satisfied solution for above equation (2.26) is either \( \frac{\partial F}{\partial y} - \frac{d}{dx} \frac{\partial F}{\partial y_x} \) or \( \eta \) equals to zero. However, \( \eta(x, y) \) is an arbitrary function, and the only left answer is:

\[
\frac{\partial F}{\partial y} - \frac{d}{dx} \frac{\partial F}{\partial y_x} = 0
\] (2.27)

and this is what we called the Euler-Lagrange Equation. Let’s rephrase it to get (2.28), where \( L_y = \frac{\partial F}{\partial y} \) and it is the partial derivatives of \( F \) with respect to function \( y(x) \). \( L_{y_x} = \frac{\partial F}{\partial y_x} \) is then partial derivatives of \( L \) with respect to \( y_x = \frac{\partial y}{\partial x} \). Using Euler-Lagrange Equation, we can find stationary functions, or extrema, that satisfy optimization problems. Those problems are the interested parts in motion planning problems by assuming paths as functions similar to \( y \) in (2.28) and length of paths is \( L_y \).

\[
L_y(x, y, y_x) - \frac{d}{dx} L_{y_x}(x, y, y_x) = 0
\] (2.28)
2.4 Euler-Lagrange with Constraints

The remaining part is how to solve problems for functional that have additional constraints on it. In most cases, it is not possible to have unlimited resources. More or less, some restrictions such as spatial or temporal limitation are possessed. For example, parking a car into a narrow space will face spatial limitation: you don’t want to hit the wall. Another constraint example is the canonical grazing animal problem. With a fixed length of rope, what shape can encircle the largest area of grazing land. Yes, the answer is a circle. But how to prove this shape is actually the best solution?

Once again, we will start from the general functional formula (2.17). In addition, we add another functional as constraint that \( y(x) \) will subject to.

\[
G(x, y, \frac{dy}{dx}) = \text{Const.} \tag{2.29}
\]

Consistent with the idea discussed in Section(2.1), our plan is creating a matching to the Lagrange multiplier \( \lambda \). As a result, combining equation (2.8), (2.17) and (2.29), we can obtain a new functional \( I[y(x)] \) in the following form:

\[
I(y) = \int_{x_1}^{x_2} (F(x, y, \frac{dy}{dx}) - \lambda G(x, y, \frac{dy}{dx})) dx \tag{2.30}
\]

where \( \lambda \) is the Lagrange multiplier because of similar reasons as (2.8). Assign (2.30) a new variable K:

\[
K(x, y, \frac{dy}{dx}, \lambda) = F(x, y, \frac{dy}{dx}) - \lambda G(x, y, \frac{dy}{dx}) \tag{2.31}
\]

Therefore, (2.30) becomes:

\[
I(y) = \int_{x_1}^{x_2} K(x, y, \frac{dy}{dx}, \lambda) dx \tag{2.32}
\]

We can use the same technique, partial differential equations, as (2.27) to find the extrema under constraints:

\[
\frac{\partial K}{\partial y} - \frac{d}{dx} \frac{\partial K}{\partial y_x} = 0 \tag{2.33}
\]

If there are needs yielding to more than one constraint, we can assign each
constraint with its own $\lambda$. Then (2.31) and (2.32) become:

$$K(x, y, \frac{dy}{dx}, \lambda) = F(x, y, \frac{dy}{dx}) - \lambda_1 G_1(x, y, \frac{dy}{dx}) - \lambda_2 G_2(x, y, \frac{dy}{dx}) - \ldots \ (2.34)$$

$$I(y) = \int_{x_1}^{x_2} K'(x, y, \frac{dy}{dx}, \lambda_1, \lambda_2, \ldots) dx \quad (2.35)$$

In conclusion, we can create a check list to solve any given functional problems:

- Define a general functional $J[f]$ as (2.17):
  $$J(f) = \int_{x_1}^{x_2} F(x, y, \frac{dy}{dx}) dx$$

- If constraint(s) need to be satisfied, depending on the number of constraints we will create a new functional $K(x, y, \frac{dy}{dx}, \lambda)$ as (2.31) or (2.34). Then choosing either (2.32) or (2.35):
  $$I(y) = \int_{x_1}^{x_2} K(x, y, \frac{dy}{dx}, \lambda) dx$$

- Solve $F(x, y, \frac{dy}{dx})$ or $K(x, y, \frac{dy}{dx}, \lambda)$ using (2.27)
  $$\frac{\partial F/K}{\partial y} - \frac{d}{dx} \frac{\partial F/K}{\partial y_x} = 0$$
Once all the mathematical tools are provided, we can start looking at the control side. The concept of control is maintaining a desired performance for a system. A system is normally composed with electrical and/or mechanical components. For example, a thermostat used to control room temperature can be defined as a system. This device can sense temperature and if the value is above desired temperature, it will turn off the air conditioner; on the contrary, if windows are opened, air conditioner will be turned on. All these works are done in order to maintain a comfortable indoor temperature. This is a typical control model that is called feedback control. In a more precise way, this kind of control uses sensors to measure the output performance and make corresponding modification based on the output to achieve a wanted behavior. Feedback control is only one of many control algorithms. In this chapter, we will review another control model called state-space control model. Admissible control, a pertinent concept to homotopy motion planning method, will also be introduced.

3.1 State-Space Control Model

The word “control” in engineering often implies applying control theory to design systems to achieve desired behaviors. For convenience, we can roughly separate the big control field into two closely related sub-fields: application and theory. Most of the time feedforward and feedback control are applied in industrial world. However, since they are unrelated to subjects in this paper, we will leave the discussion about them out. While state-space control model is the footstone for many theoretical control research, and in the remaining parts of this section, we will review some basics about this model.

Generally, a state-space control model has the following form:
\[ \dot{x}(t) = A(t)x(t) + B(t)u(t) \quad (3.1) \]
\[ y(t) = C(t)x(t) + D(t)u(t) \quad (3.2) \]

where variable matrices: state \( x(t) \), input \( u(t) \) and output \( y(t) \) are all correspond to time in the form of:

\[
\text{state } x(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{pmatrix} \in \mathbb{R}^n
\quad (3.3)
\]

\[
\text{input } u(t) = \begin{pmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_m(t) \end{pmatrix} \in \mathbb{R}^m
\quad (3.4)
\]

\[
\text{output } y(t) = \begin{pmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_p(t) \end{pmatrix} \in \mathbb{R}^p
\quad (3.5)
\]

Furthermore, each matrix \( A(t), B(t), C(t) \) and \( D(t) \) is given a nomenclature based on their functionalities in the model. These are matrices with the appropriate dimensions satisfy sizes of state variables:

\[
A(t) : n \times n \quad \text{System Matrix}
\]
\[
B(t) : n \times m \quad \text{Input Matrix}
\]
\[
C(t) : p \times m \quad \text{Output Matrix}
\]
\[
D(t) : p \times m \quad \text{Feedthrough Matrix}
\]

According to the above model, it is easy to get clues about future states based on current states and inputs. For now, let’s denote \( A(t) \), the system matrix, be a zero matrix. This simplification allows next state of the system becomes independent of current states and solely depends on input
information. Therefore, we can rewrite state-space model as:

\[
\dot{x}(t) = (B_1 \ B_2 \ \ldots \ B_m) \begin{pmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_m(t) \end{pmatrix} \tag{3.6}
\]

This model can be used to demonstrate a simple car movement. Assume we have a car such that at beginning \( t = 0 \), the car is at the origin \( x(t = 0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}^T \). After 1 second, we want the car to appear on desired position \( x(t = 1) = \begin{pmatrix} 1 \\ 2 \end{pmatrix}^T \), where the first value in matrices represent x-position and the second ones are y-position. Without considering the mechanics behind vehicle dynamics and assuming the car can always move in any directions, which is of course not feasible in real world, we can find a path that connects the starting and final states can be designed using state space model. The following inputs \( u_1 \) and \( u_2 \) in (3.7) can be visualized as taking one step forward on positive x and y directions.

\[
x(t) = \begin{pmatrix} 1 \\ 0 \end{pmatrix} u_1 + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_2 \tag{3.7}
\]

If the system is in discrete time domain, the solution is trivial: replace \( u_1 = 1 \) and \( u_2 = 2 \).

\[
x(t = 1) = 1 \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} + 2 \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \tag{3.8}
\]

Yet, in real world, time is continuous and therefore, integral needs be employed here and we set \( T = 1s \) in (3.9):

\[
x(t = 1) = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \int_0^{T=1s} u_1(t)dt + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \int_0^{T=1s} u_2(t)dt \tag{3.9}
\]

From this example and because of our omission of system matrix, we find that finding next state depends only on input values \( u(t) \). Therefore, we can somehow come up a way to integrate all input variables, which is the admissible control.
3.2 Admissible Control

It is the time to give a formal definition of admissible control. Suppose there exists a path \( P(t) \) with known starting and final states \( P(t_0) = x_0 \) and \( P(t_1) = x_1 \). It is worth mentioning that this path not only exists in real world as we normally think, but also in configuration space: a space consisted by all degrees of freedom for a given system. Defining a path as an *admissible path* if next path state \( \dot{P}(t) \) is in the span of all input matrices for a given system, shown as equation (3.10):

\[
P(t) : \{ \text{admissible if } \dot{P}(t) \in \text{Span}\{B_1(P(t)), B_2(P(t)), ..., B_n(P(t))\} \}
\]

In other words, once state \( P(t) \) is in the intersection of subspaces consisted by \( B_1, B_2, ... B_n \), it is a candidate for final state: once reasonable input variables are given, a path that can connect starting state \( P(t_0) \) and \( P(t) \) exists.

Let’s see the unicycle example. An unicycle shown as Fig.3.1 can only walk along its wheel direction. Additionally, the unicycle has three degrees of freedom: \((x, y, \theta)\). First two variables are positions of the unicycle on floor and \( \theta \) is the wheel rotation angle. For this reason, the configuration space of this model also has three dimensions. Hence, the derivative of path state

![Unicycle diagram](image)

Figure 3.1: Unicycle with center mass \((x, y)\) and wheel rotation angle \(\theta\)
\( \dot{P}(t) \), which also is the velocity vector, of this model can be expressed as:

\[
\dot{P}(t) := \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \\ 0 \end{pmatrix} u_1 + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} u_2 \tag{3.11}
\]

The first matrix after the second equal sign is the position matrix that relates with changes in x,y direction. It is reasonable that position is represented as sinusoidal functions of angle \( \theta \) due to restriction of headed direction. While second column matrix is the changing of rotation angle of wheel.

A way to verify whether a given path is admissible for any systems is building a matrix contains all input matrices \( B_i(t) \) and the state \( \dot{P}(t) \) we want to examine. Then by calculating the determinant of this new matrix we can find the satisfied value \( \theta \) or fail if such solution does not exist. For unicycle system, we want to check \( P(t) = \begin{pmatrix} t^2 \\ t^2 \\ t \end{pmatrix} \) is achievable from starting position, the origin, or not. Our first step is chasing the definition of admissible by finding the derivative of path with respect to time \( t \):

\[
P(t) = \begin{pmatrix} t^2 \\ t^2 \\ t \end{pmatrix} \Rightarrow \dot{P}(t) = \frac{dP(t)}{dt} = \begin{pmatrix} 2t \\ 2t \\ 1 \end{pmatrix} \tag{3.12}
\]

Then appending (3.12) to those input matrices in (3.11) and we denote the new matrix as \( A \):

\[
A = \left[ B_1(P(t))|B_2(P(t))|\dot{P}(t) \right] = \begin{pmatrix} \cos(\theta) & 0 & 2t \\ \sin(\theta) & 0 & 2t \\ 0 & 1 & 2t \end{pmatrix} \tag{3.13}
\]

Based on the property of linearly independence, we can check the independence of columns in a given matrix by finding the determinant of the whole matrix. If the determinant equals to zero, \( \det(A) = 0 \), then it implies the last column(\( \dot{P}(t) \)) can be expressed as linear combination of previous columns,which means \( \dot{P}(t) \) is in the span consisted by all input matrices.
Based on this knowledge, the next step is facile:

\[
\det(A) = \det \begin{pmatrix}
\cos(\theta) & 0 & 2t \\
\sin(\theta) & 0 & 2t \\
0 & 1 & 2t
\end{pmatrix} = 0
\] (3.14)

\[-2t \cdot \cos(\theta) + 2t \cdot \sin(\theta) = 0\] (3.15)

Solving above equation will result:

**Solution:** \( \theta = \frac{\pi}{4} + n\pi \) where \( n \in \mathbb{Z} \) (3.16)

This solution informs us that once we can assure the input variables are always equals to \( \theta \), the unicycle will eventually achieve \( P(t) = \left( t^2 \ t^2 \ t \right)^T \).
CHAPTER 4

HOMOTOPY METHOD FOR MOTION PLANNING

In previous chapters, we derived related formulas to find optimization solutions for functionals under constraints in Section(2.3) and went through the concept of admissible control Section(3.2). These are two tools conducive to finding method solving motion planning problems. For motion planning designing of a given system, such as a car or a humanoid robot, a control algorithm will allow it interacting with the environment and to perform tasks. At the same time, the system can seek optimizations to minimize certain factors: distance, time or some other limited elements. Motion planning designs the path in both real world and configuration space. This path allows system successfully accomplishing all tasks and seeking optimization simultaneously. Control then can ensure the machine will move in the designated way, without accidents. Since previous chapters provide all essentials tools, it is time to explore this homotopy method. However, before going straight to the implementation of this method, we will examin different kinds of constraints. Therefore, in this chapter, holonomic, non-holonomic and avoidance obstacle constraints will be defined first. Metric $G$ will be proposed to assemble all these constraints. Finally, we will inspect in details about the method.

4.1 Constraints and Metric G

When talking about constraints, the most intuitive idea of what they are might be obstacles along the path. However, the definition of constraints is more extensive than this. In our cases, there are three types of constraints: holonomic, non-holonomic and obstacle constraints. For the first two constraints, a straightforward difference between them is that holonomic constraints reduces the number of degree of freedom of a system such that constraints are applied to restrict position in configuration space. However,
**nonholonomic constraints** only specify limitations on velocity but not on the position and it doesn’t reduce any degrees of freedom for a system.

For a ball that can be placed everywhere, we can use \((x,y,z)\) to represent its location in any three dimensional space. Therefore, the ball has three degrees of freedom. However, if the ball is placed on a table such that it is only allowed to move along the table surface, then a holonomic constraint is imposed on it because this ball loses its \(z\)-directional degree of freedom. While unicycle mentioned in Section(3.2) is an example of **non-holonomic constraint**. Even though certain moving directions are prohibited for a unicycle: it can’t move in the perpendicular direction to the wheel rotational direction, shown as (4.1), such constraints don’t reduce the degrees of freedom. All variables of the unicycle can still be changed: degrees of freedom are still three.

\[
\dot{x} \sin \theta - \dot{y} \cos \theta = 0 \quad (4.1)
\]

In order to incorporate these two types of constraints in the method, a Riemannian metric \(G\) is bringing in. The metric \(G\) is in positive-definite bilinear form and we define \(G\) as:

\[
G(x) = F(x)\text{diag}([k,...,k 1...1])F^T(x) \quad (4.2)
\]

The \(F(x)\) is composed of two sub-matrices and constructed as (4.3). The first submatrix \(F_c \in \mathbb{R}^{n \times p}\) is the one that embodies holonomic and non-holonomic constraints. By convention, holonomic related elements are placed in top rows while non-holonomic are placed at bottom. \(F_f \in \mathbb{R}^{n \times (n-p)}\) satisfies \(F_f^TF_c = 0\), that is \(F_f\) is in the null space of \(F_c\). In another way: this matrix represents the undesirable or prohibited moving directions of all degrees of freedom in configuration space. Thus, \(F(x)\) is:

\[
F(x) = \begin{bmatrix} F_c \\ F_f \end{bmatrix} \quad (4.3)
\]

Therefore, we obtain a square matrix \(F(x) \in \mathbb{R}^{n \times n}\).

Since these constraints are undesired directions in configuration space, a cost variable \(k\) will be used to add cost for them by setting \(k\) as large as possible, e.g., \(k = 150\). With the implementation of diagonal matrix such that those rows corresponding to constraints will have cost values \(k\), we can

---

21
ensure when algorithm tries to find a path, it will not go to those unwanted directions because their costs are larger than desired ones. Furthermore, diagonal matrix can allow us access each constraint independently and choose different feedbacks correspondingly. The remaining columns of diagonal matrix are all set to 1 to work with $F^T(x)$.

The last relevant constraint is obstacle avoidance constraints, which are ones we have intuitions on them. They are constraints at the possible path of a system but are prohibited by environment. For such constraints, a barrier function $r(x)$ is defined to apply same strategy as holo- and nonholo-constraints: adding cost to undesired direction. $r(x)$ is a function related with the distance between system and obstacles. If $d(x)$ denotes the distance between system and obstacles, then $r(x) = \frac{1}{d(x)}$ is a good candidate for barrier function. The value of $r(x)$ will be very large or even unbounded when the distance is very small: $d(x) \ll 1$. Otherwise, it is a normal value that will not affect the behavior at all. $r(x)$ and $k$ are both assistive tools to help us avoiding those undesirable path directions. Therefore, the final version of metric matrix $G$ is:

$$G = r(x)F(x)\text{diag}([k...k 1...1])F^T(x) \quad (4.4)$$

We will use a robot leg model. In configuration space, this model have five variables: $\begin{pmatrix} x & y & \theta_1 & \theta_2 & \theta_3 \end{pmatrix}^T \in \mathbb{R}^5$. Where $x$ and $y$ represent the position of leg's end effector, $\theta_1$ is the torso angle and $\theta_2$ is the angle between torso and upper legs while the last $\theta_3$ is the lower leg angle. Let $L_1, L_2, L_3$ be the length of torso, upper and lower legs, respectively. The leg model is fixed at a starting point: $(x_0, y_0) = (0, 0)$.

Our first step is writing down all holonomic and non-holonomic constraints. In our case, we only have holonomic constraints:

$$q_1(x) := x = L_1 \cos \theta_1 - L_2 \cos(\theta_1 + \theta_2) + L_3 \cos(\pi - \theta_1 - \theta_2 - \theta_3) \quad (4.5)$$

$$q_2(x) := y = L_1 \sin \theta_1 - L_2 \sin(\theta_1 + \theta_2) - L_3 \sin(\pi - \theta_1 - \theta_2 - \theta_3) \quad (4.6)$$

Then we can find the differential of above equations with respect to all degrees
of freedom will obtain (4.8) and (4.10):

\[
q_1(x) = \left[ \begin{array}{c} \frac{\partial q_1}{\partial x} \frac{\partial q_1}{\partial y} \frac{\partial q_1}{\partial \theta_1} \frac{\partial q_1}{\partial \theta_2} \frac{\partial q_1}{\partial \theta_3} \\
-1 \ 0 \ -L_1 \sin_1 + L_2 \sin_{12} + L_3 \sin_{123} \ L_2 \sin_{12} + L_3 \sin_{123} \ L_3 \sin_{123} \end{array} \right] \tag{4.7}
\]

\[
q_2(x) = \left[ \begin{array}{c} \frac{\partial q_2}{\partial x} \frac{\partial q_2}{\partial y} \frac{\partial q_2}{\partial \theta_2} \frac{\partial q_2}{\partial \theta_3} \\
0 \ -1 \ L_1 \cos_1 - L_2 \cos_{12} - L_3 \cos_{123} \ -L_2 \cos_{12} - L_3 \cos_{123} \ -L_3 \cos_{123} \end{array} \right] \tag{4.8}
\]

where \( \cos_1 := \cos(\theta_1) \), \( \cos_{12} = \cos(\theta_1 + \theta_2) \) and so on. The constraint matrix \( F_c \) and correlative \( F_f \) will be:

\[
F_c(x) := \left[ \begin{array}{ccc}
1 & 0 & 0 \\
\sin_1 - \sin_{12} - \sin_{123} & \cos_1 + \cos_{12} + \cos_{123} \\
- \sin_{12} - \sin_{123} & \cos_{12} + \cos_{123} \\
\sin_{123} & L_3 \cos_{123} \\
\end{array} \right] \tag{4.11}
\]

\[
F_f(x) = \left[ \begin{array}{ccc}
\sin_{123} + \sin_{12} - \sin_1 & \sin_{123} + \sin_{12} & \sin_{123} \\
\cos_1 - \cos_{12} - \cos_{123} & -\cos_{123} - \cos_{12} & -\cos_{123} \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{array} \right] \tag{4.12}
\]

\( F_f \) is found using techniques about null space. We construct \( F(x) \) by appending \( F_f \) to \( F_c \):

\[
\left[ \begin{array}{ccccccc}
1 & 0 & \sin_{123} + \sin_{12} - \sin_1 & \sin_{123} + \sin_{12} & \sin_{123} \\
0 & 1 & \cos_1 - \cos_{12} - \cos_{123} & -\cos_{123} - \cos_{12} & -\cos_{123} \\
\sin_1 - \sin_{12} - \sin_{123} & \cos_1 + \cos_{12} + \cos_{123} & 1 & 0 & 0 \\
- \sin_{12} - \sin_{123} & \cos_{12} + \cos_{123} & 0 & 1 & 0 \\
\sin_{123} & L_3 \cos_{123} & 0 & 0 & 1 \\
\end{array} \right] \tag{4.13}
\]

Due to space limitation, the full form of \( G(x) \) will not be shown here. But as
mentioned before, $G(x)$ can be easily calculated by combining (4.13) using (4.4) with software help such as MATLAB. The final form of $G(x)$ will contain costs $r(x)$ and $k$.

4.2 Homotopy Method

Before finding the solution, the first thing we need to do is to figure out what is the problem. In our motion planning designing, we define path $p(t) : [0, 1] \rightarrow C$ as a continuously differentiable function in configuration space. Moreover, there are two path functions $p_1(t)$ and $p_2(t)$ such that they share common fixed end points: $p_0(t = 0) = p_1(t = 0)$ and $p_0(t = 1) = p_1(t = 1)$. The general idea of this method is claiming both functions are in topological space and we will find a way to continuously deform one path to another one, such deformation is called homotopy between the two functions, which is a concept in topology: the study of geometry or space such that their properties are maintained after deformation.

Furthermore, recall the definition of Euclidean norms and we will define the length of path $L(p)$ as functional from this definition:

$$L[p(t)] := \int_{t=0}^{t=1} \sqrt{||\dot{p}(t)||} dt = \int_{0}^{1} \sqrt{\dot{p}(t)^T G \dot{p}(t)} dt$$

(4.14)

where $G$ is the metric we discussed in Section(4.1). Because extrema behaviors will not change with or without square root symbol, it is safe to take the square root symbol out from (4.14) to get (4.15): the energy functional.

$$E[p(t)] := \int_{0}^{1} \dot{p}(t)^T G \dot{p}(t) dt$$

(4.15)

Then taking partial differential of (4.15) in a similar manner to (2.33) will give us the geometric heat flow, or gradient flow, equation:

$$\frac{\partial}{\partial s} v_i(s, t) = \frac{\partial^2}{\partial t^2} v_i(s, t) + \sum_{j,k} \Gamma^i_{jk} \frac{\partial v_j}{\partial t} \frac{\partial v_k}{\partial t}$$

(4.16)
where symbol $\Gamma^i_{jk}$ is:

$$
\Gamma^i_{jk} := \frac{1}{2} \sum_{l=1}^{n} g^{il} \left( \frac{\partial g_{lj}}{\partial x_k} + \frac{\partial g_{lk}}{\partial x_j} - \frac{\partial g_{jk}}{\partial x_l} \right) \tag{4.17}
$$

$g_{ij}(x)$ is the $ij^{th}$ entry of inverse of metric: $G^{-1}(x)$ and $v(s, t)$ is a continuous function in topological space such that $v(0, t) = p_0(t)$, which is the intial path.

The boundary conditions for $v(s, t)$ are: $v(s, 0) = p_0(s)$ and $v(s, 1) = p_0(1)$. Thus, the final solution can be found as:

$$
p_1(t) = \lim_{s \to \infty} v(s, t) \tag{4.18}
$$

Based on definition of (3.10), corresponding control input varaibles can be derived by setting $\bar{u}(t)$ as:

$$
\bar{u} = B^\dagger F f F^\dagger f \dot{p}(t) \tag{4.19}
$$

$\dagger$ is the symbol for sudo-inverse . The final trajectory of the system will be (4.20) where $b_i(x^*)$ is the control input correspond to $i^{th}$ degree of freedom variable:

$$
\dot{x}^*(t) = \sum_i \bar{u}_i b_i(x^*) \tag{4.20}
$$

Thus, if the obtained path from (4.18) is admissible, then $p(t)$ equals to $x^*(t)$. 
In this last chapter, implementations of two examples will be examined: One is the three degrees of freedom unicycle model described in Section 3.2 and a more complex half-body quadrupedal animal model.

The implementation of this method is relatively easy and straightforward. There are general procedures that can be followed. What we need to provide is an initial path $p_0(t)$. This path does not need satisfy any holonomic and non-holonomic constraints. However, it should avoid obstacles constraints. Such path is easy to find. Using the unicycle model as an example. The goal in this scenario is parking itself into a parking lot shown as fig. 5.1. The red lines are avoidance obstacles and the width of available parking lot is unit 2 as shown. A simple initial path $p_0(t)$ is the black triangular shape curve in Fig. 5.1.

![Figure 5.1: Initial path for a unicycle with avoidance obstacles](image)

After running this method, we will obtain a desired and satisfied path $p_1(t)$ shown in Fig. 5.2. This path not only successfully keeps away from all avoidance obstacles, the red walls, but also satisfies holonomic and non-holonomic constraints. If we run the simulation of the unicycle to follow the path $p_1(t)$, we will see the movement of the unicycle is reasonable and matches the physical laws.

Another example is a robotic leg that similar to the model mentioned in
Section (4.2). However, this model is more complex and has nine degrees of freedom:

\[
\dot{x}(t) = \begin{pmatrix} x_1 & y_1 & x_2 & y_2 & \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 \end{pmatrix}^T
\]  

(5.1)

where tuple \((x_1, y_1)\) is the position of rear leg end effector and \((x_2, y_2)\) is for frontal leg. Akin to model in Section (4.2), \(\theta_1\) and \(\theta_3\) are angles between torso and upper legs while \(\theta_2\) and \(\theta_4\) are angles between upper and lower legs. Finally, as before, \(\theta_5\) is the torso angle. However, due to the complexity of this model, we will just show some snapshots of the final path in Fig. 5.3 and Fig. 5.4.

The easiness to implement this method demonstrates its usage pos-
sibilities in practical applications. In addition, the calculation times for both model are relatively quick, which further show the hope to utilize such method in reality.
REFERENCES


