An Overview of the Class of Rapidly-Exploring Random Trees

B.Sc. Artificial Intelligence - 7.5 ECTS

Bryan Cardenas Guevara
First Advisor: dr. G.A.W. Vreeswijk
Second Advisor: dr. Benjamin Rin

Faculty of Humanities
Utrecht University

July 9, 2018

AN OVERVIEW OF THE CLASS OF RAPIDLY-EXPLORING RANDOM TREES

Bryan Cardenas Guevara
5683793

7.5 ECTS thesis submitted in partial fulfillment of a
Baccalaureus Scientiae degree in Artificial Intelligence

Advisor
dr. G.A.W. (Gerard) Vreeswijk
dr. Benjamin Rin

First Thesis Advisor
dr. G.A.W. (Gerard) Vreeswijk

Faculty of Humanities
Artificial Intelligence
Utrecht University
Utrecht, July 2018

An Overview of the Class of Rapidly-Exploring Random Trees
The Value and Purpose of Rapidly-Exploring Random Trees
7.5 ECTS thesis submitted in partial fulfillment of a B.Sc. degree in Artificial Intelligence

Faculty of Humanities
Artificial Intelligence
Utrecht University
Trans 10, 1.05A
3512 JK Utrecht, Utrecht
The Netherlands

Telephone: 030 253 6096

Bibliographic information:
Bryan Cardenas Guevara, 2018, An Overview of the Class of Rapidly-Exploring Random Trees, B.Sc. thesis, Faculty of Humanities, Utrecht University.

Utrecht, The Netherlands, July 2018

Contents

Abstract

In motion planning the objective is to move a robot from one place to another in a possibly constrained environment. There exist exact methods to solve this problem but exact solutions in constrained higher dimensional spaces are in general unfeasible. It has been shown that sampling-based algorithms, such as the Rapidly-Exploring Random Trees (RRT), perform well when solving motion planning problems under certain constraints. A tree is constructed incrementally to search for the goal by taking samples from the state space. The class of RRTs has been shown to be probabilistically complete; it will eventually find a solution as the tree grows. The basic RRT algorithm has been shown to be asymptotically suboptimal, it will almost surely converge to a path of suboptimal cost. RRT* was proposed to overcome this problem and was in turn shown to be asymptotically optimal. Informed RRT* exploited this property by introducing a heuristic that focused the algorithm between the start and the goal. In this paper the theory behind these baseline algorithms, as well as some of their variants, is explained along with their behaviour and properties. Extensive research has been done in the field of RRTs but no updated account exists on the class of RRTs. This paper organises an overarching review of the class of RRTs. Additionally, the three baseline algorithms are compared to each other. From the analysis of this comparison it follows that Informed RRT* converges significantly faster to optimal paths, as opposed to RRT* and RRT. Finally, current issues and assumptions for future research are discussed.

Keywords - Motion Planning Algorithm; Sampling-Based Planning; Rapidly-Exploring Random Trees; Robotics; RRT*; Informed RRT*; Path Optimality.

1. Introduction

In artificial intelligence, graph-search techniques have been used to solve discrete tasks such as moving blocks or solving a Rubik's cube. The A* algorithm (P. E. Hart & Raphael, 1968) is an example of an exact graph search technique, since it is able to exactly solve a problem given a discrete representation of the problem. It will always find a solution, if the solution exists. These kind of graph-search (discrete) algorithms are also able to find solutions in continuous spaces. The continuous space has to be discretised, the quality of discretisation of the space results in a better solution but at the cost of computation time and space (Bertsekas, 1975). Furthermore, as the dimensions are increased, the discrete representation chosen for the problem grows exponentially. In a motion planning problem, however, the task is to solve a problem in a continuous space without an explicit discrete representation of the space. The motion planning problem has also been called the Piano-Mover's Problem (Schwartz & Sharir, 1983) where the task is to move a piano through a space in which obstacles reside. The problem is to get the piano (a robot) from point A to point B, while avoiding these obstacles.

Since the Piano-Mover's Problem has been proven to be PSPACE-hard (J.H.Reif, 1979), employing exact methods on complex planning problems in continuous space could become unfeasible because these do not scale well and are usually bounded to two-dimensional

problems (LaValle, 2006). Techniques that use notions from probability theory such as the so called sampling-based motion planning algorithms avoid the problem of constructing a discrete representation by directly drawing samples from the state space. Although these methods are not exact, some of these are proven to be probabilistically complete. That is, given enough time, they will find a solution with a certain probability. Developments in robotics and motion planning have seen the introduction of such algorithms. The invention of Probabilistic Roadmaps has shown that motion planning in continuous spaces could become feasible (Kavraki et al., 1996; Overmars, 1998). More recently, Rapidly-Exploring Random Trees (RRT) (LaValle, 1998) have been shown to be effective in higher dimensional spaces and with complex constraints on the problem domain (LaValle & Kuffner, 2000). RRT based algorithms construct a space-filling tree by taking samples from the space to find a goal, while considering the constraints in the motion planning problem. RRT showed promising results in a wide variety of problems such as robotic arms, moving pianos and docking manoeuvres for satellites (LaValle & Kuffner, 2000). RRT is not restricted to the field of Robotics or path planning, it also serves as a modelling tool for Diffusion Limited Aggregation (DLA) (Witten & Sander, 1981; LaValle & Kuffner, 2000) where particles perform what seems to be a random walk constructing a tree similar to the tree generated by RRT. Despite of these results, RRT was proven to be asymptotically suboptimal (Karaman & Frazzoli, 2010) and unfit for some problems where an optimal solution was needed. Therefore, RRT* was proposed (Karaman & Frazzoli, 2010; 2009) to overcome this problem and was proven to be symptotically optimal. Since the introduction of RRT*, many state of the art variations have shown up; RRT*-Smart, Informed RRT*, RABIT* and FMT* are robust in high dimensional spaces with complex constraints including motion planning for helicopters, robot arms, humanoids and self-driving cars but also for drug design (Nasir et al., 2013; Choudhury et al., 2016; Gammell et al., 2014b; L. Janson & Pavone, 2015; Kim & Song, 2018). The algorithms of RRT, RRT* and Informed RRT* will be regarded as baseline algorithms that lay the groundwork for many of their variations.

The development the class of RRTs has been accelerating since 2010 and is increasingly being used in robotic tasks. However, since it concerns more recent developments, there is no readily available and compact literature charting the important features behind many of the proposed algorithms. This way, much of the theory and implementations of RRT are only accessible for academia. Moreover, there does not seem to be a consensus regarding some of the notation and the provided methods could differ across papers. This in turn means that an implementation could become more difficult and that basic concepts become harder to understand.

To the best of our knowledge, no book, survey or academic article exists where an updated account of the theory behind the base algorithms is summarised. In (LaValle, 2006), Steven M. LaValle brings together a substantial amount of theory behind motion planning but also sampling-based algorithms such as RRT. Yet, it does not include recent and updated developments beyond the regular algorithm for RRT.

Therefore, this paper is devoted to provide such a summary. The theory behind the behaviour of RRTs is discussed along with an overview of their algorithmic implementation and some variations improving on their underlying concepts. Their behaviour within simple planning domains is also discussed and is shown in order to provide an ease of understanding. Furthermore, the base algorithms of RRT, RRT* and Informed RRT* are compared to

each other in two simple two-dimensional planning domains; one with no obstacles and one with obstacles with the optimal path going through a narrow passage. The algorithms are run Monte Carlo style and their rate of convergence is examined.

The sampling based nature of RRT allows for a rapid exploration on the two planning problems. RRT finds a solution to the problem but quickly converges to a local minimum yielding a suboptimal path. Furthermore, the RRT algorithm could not find a solution through the narrow passage in the domain with obstacles. RRT* demonstrates a clear improvement by converging to the optimal path but takes significantly longer to do so. Informed RRT* outperforms RRT*'s rate of convergence by concentrating on the paths between the start and the goal. On the two considered problems Informfed RRT* showed an order-of-magnitude improvement. Both RRT* and Informed RRT* eventually found the optimal path through the narrow passage.

This paper is organised as follows. The Background section (2) provides key concepts for the algorithms and will also lay the groundwork for the notation and some formal definitions that will be used throughout this paper. Section 3 discusses three key algorithms and provides an understanding for their behaviour, while also presenting some of their properties and variants. Section 4 on the simulations provides a comparison between the three base algorithms and an analysis with two simple planning problems. Section 5 presents a general discussion which contains some concluding remarks

2. Background

2.1. Notation and Preliminary Material

Prior to the introduction of the algorithms some preliminary material is presented concerning some notation and problems specific to the algorithms. Note that throughout this paper we will assume holonomic environments, which will be explained later in Section 2.6 on kinodynamics and holonomy. Moreover, we assume static environments, we will not consider dynamic or partially observable environments. It is also assumed that the reader has basic knowledge of probability theory and linear algebra.

Before we jump to the explanation of the algorithms, we will clarify some notation that will be used throughout this paper. The notation used in this paper is mostly consistent with the notation used in the literature.

In this paper $\|\cdot\|$ represents the Euclidean norm. We will also denote $\text{diag}\{\cdot\}$ and $\det(\cdot)$ as the diagonal and the determinant of a matrix respectively.

Furthermore, given a set $X \subseteq \mathbb{R}^d$, a directed graph G is a tuple (V, E) where $V \subseteq X$ is the set containing the vertices (nodes) of the graph (tree) and E is the set containing the edges, with $E \subseteq V \times V$. A tree is a directed graph where each vertex has exactly one parent except for the root vertex (x_{init}) which has no parent. In this paper we will refer to the vertices in the tree as nodes.

Moreover, we denote a path in the tree from a starting node q_{start} in the path to an-

other node q_{arb} as σ[1]. Given two paths σ_1 and σ_2, the two paths could be concatenated and this concatenation is denoted as $(\sigma_1 \mid \sigma_2)$. Let $\Sigma_{X_{free}}$ be the set of all paths that are non-trivial, i.e., all paths have a length strictly greater than zero.

A cost function $c : \Sigma_{X_{free}} \mapsto \mathbb{R}_{\geq 0}$ assigns a cost to a path from $\Sigma_{X_{free}}$.

Given a radius $r > 0$ with a point $x \in X$, the r-ball area $B_{x,r}$ centered at x is defined as $B_{x,r} = \{y \in X \mid \|y - x\| \leq r\}$.

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space. A random variable Y is a function $Y : \Omega \mapsto \mathbb{R}$. A sequence of random variables $\{Y_i\}_{i \in \mathbb{N}}$ has an almost sure convergence if $\mathbb{P}\{lim_{i \to \infty} Y_i = Y\} = 1$ and a sure convergence to a random variable Y if $lim_{i \to \infty} Y_i(\omega) = Y(\omega)$ for all $\omega \in \Omega$. A motion planning algorithm is asymptotically optimal if $\mathbb{P}\{lim_{i \to \infty} Y_i = c^*\} = 1$ and, conversely, is asymptotically suboptimal if $\mathbb{P}\{lim_{i \to \infty} Y_i = c^*\} = 0$. That is, in this paper and throughout RRT literature, an algorithm is asymptotically optimal to the extent that it manifests an almost sure convergence to an optimal solution as the number of iterations approaches infinity.

There are some primitive procedures (or functions) that the algorithms use. These are,

- *Sample*, that gets a point $x \in X_{free}$ from a uniform distribution[2]
- *Steer*, that given two points (nodes), and given ϵ, $x, y \in X_{free}$ returns a point $z \in \mathbb{R}^d$ such that $\|z - y\|$ is minimised and $\|z - x\| \leq \epsilon$, i.e., z must be chosen in the direction of y while the distance between z and x is bounded by the specified step-size $\epsilon > 0$. Usually, the point x is a node from the tree and y is a sampled point, x being the nearest neighbour of y. The steer function is therefore defined by:

$$Steer(x, y) = argmin_{z \in \mathbb{R}^d, \|z-x\| \leq \epsilon} \|z - y\|.$$

- *Nearest*, that given the set V of nodes and a point $x \in X_{free}$ returns a node $v \in V$ that is closest, or *nearest*, to x. Usually, the Euclidean distance is used as the distance function, although other distance functions are possible. In this paper we assume that the *Nearest* function is defined as:

$$Nearest(V, x) = argmin_{v \in V} \|x - v\|$$

- *ObstacleFree*, that given two points (nodes) $x, y \in X_{free}$ the path, or line segment in the case of a Euclidean distant measure, between these two points is checked for obstacles.

These four procedures are essential for the foundation of each algorithm discussed in the following sections.

[1] In (Karaman & Frazzoli, 2010) σ is a continuous function mapping the length of a path to a path in X, i.e., the state space

[2] The sampling distribution in RRTs is usually uniformly distributed, although there have been cases where some other distribution is used (D. Kim & e. Yoon, 2014; Jouandeau & Cherif, 2004).

2.2. Problem Formulation

A motion planning problem for RRT, RRT* and Informed RRT, is usually characterised by the following description:

1. The State space is $X \subseteq \mathbb{R}^d$, where d is the dimension of the state space. X consists of X_{free} which denotes the search space free of obstacles and $X_{obstacle}$ which denotes the space occupied by obstacles.
2. $x_{init} \in X_{free}$ is the starting point of the RRT, $X_{goal} \subseteq X_{free}$ is the goal area the algorithm searches for. Although the goal area could be a single point in X_{free}, it is favourable to include a small area around the goal.
3. $\epsilon > 0$, is a predefined step-size that denotes the maximum distance between nodes in the search tree.
4. Let K denote the maximum number of iterations for an RRT.
5. Problem 1: *Feasible Path Planning*. The purpose of the algorithm is to find a path σ, such that $\sigma(0) = x_{init}$ and $\sigma(s) \in X_{goal}$ if one exists.
6. Problem 2: *Optimal Path Planning*. The purpose of the algorithm is to find a path $\sigma*$ with minimum cost c between x_{init} and X_{goal} such that $c(\sigma*) = min\{c(\sigma_f) \mid \sigma \in \Sigma_{X_{free}}\}$.

2.3. Sampling-Based Motion Planning Algorithms

This subsection will provide a short introduction on sampling-based algorithms that address problems four and five that were stated in Section 2.2 with a sampling scheme.

The general idea is to search the state space by drawing samples from this space. It is important to note that sampling-based motion planning algorithms regard the obstacle space as unknown. It does not try to search through the state space with an explicit representation of the space, i.e., it searches without an explicit construction of the obstacles. The state space gets sampled, instead of being computed explicitly. Although sampling-based algorithms are generally not considered complete, some are probabilistically complete. An algorithm is said to be complete if it successfully returns a solution in finite time or, if a solution does not exist, successfully reports that there is no such solution. An algorithm is said to be probabilistically complete when the probability of finding a solution converges to one as the number of iterations grows to infinity. Combinatorial motion planning algorithms are an example of complete algorithms (Latombe, 1991; Hossam & David, 1981; LaValle, 2006) that have the guarantee to find a solution or show that a solution does not exist. These kind of algorithms are also referred to as being exact algorithms. In contrast, sampling-based algorithms will usually run forever if a solution does not exist. Combinatorial algorithms must explicitly encode the state space to construct a roadmap[3] in order to explore the space for a solution. However, motion planning problems are generally PSPACE-hard (J.H.Reif, 1979), which implies NP-hard, thus an exact solution could be unfeasible. Nevertheless, there exist

[3]A roadmap is essentially a graph in the state space to be used for exploration

combinatorial algorithms that can partition the state space in linear time (Chazelle, 1991) but these are generally very impractical, too difficult to implement efficiently or do not scale well when the number of dimensions is increased.

A well known sampling-based motion planning algorithm is the probabilistic roadmap (PRM) (Kavraki et al., 1996). This method handles multiple queries by constructing a roadmap in the state space, meaning that there is not just one but several starting points from where the queries are issued. The nodes of the roadmap (graph) are uniformly sampled during what is called the preprocessing phase. A node is only added to the graph if the sampled point does not lie within the obstacle space; it does this incrementally[4]. During the query phase each pair of nodes sampled is connected to form the edges in the graph. This approach found its success mainly because it is robust in high-dimensional search spaces. It was in fact the first to solve motion planning problems with more than four dimensions. However, PRM was initially designed for holonomic environments (Section 2.6) and requires thousands of connections between the nodes due to its multiple query nature. Hence a new class of sampling-based motion planning algorithms had to be invented.

2.4. The Class of Rapidly-Exploring Random Trees

A Rapidly-Exploring Random Tree (RRT) is a single query sampling-based motion planning method; RRT has one starting point from which the state space is searched and previous queries are not considered. A single-query algorithm tries to answer each query from scratch. The basic algorithm was specifically designed to be capable of incorporating differential constraints and manage non-holonomic environments, which will be discussed in the next subsection. A key concept is that a random sequence is probably dense. A sequence of samples is said to be dense when the sequence comes arbitrarily close to any other sequence when it is sampled in the limit. Thus, an infinite sequence of sampled points will be dense in their given space. The trees that are constructed are akin to space filling curves, when the number of samples drawn approach infinity. Space filling curves are fractals that densely cover its space first discovered by Giuseppe Peano in 1890 (G., 1890). However, RRT consists of shorter paths and from each node in the tree there exists a path to the initial, or root, node of the tree. In the limit, RRT will densely cover the state space, hence the class of RRT is sometimes alternatively referred to as Rapidly-Exploring Dense Tree (RDT). The state space is filled incrementally by sampling nodes from the space and connecting these to their nearest neighbour in the tree. As we will see in the section on algorithms, there is a variety of modifications to this algorithm giving rise to several different variants that try to improve on some short-comings that we will also encounter along the way. Generally, the family of RDT follow the next steps:

1. **Initialisation:** A graph is initialised with a set of nodes containing just the starting point and an empty set of edges.
2. **Node Selection:** A node is usually selected with a nearest neighbour procedure.
3. **Planning Method:** Typically, when a node has been selected, it is checked whether the defined constraints of the motion planning problem are satisfied, moreover a collision

[4] With each iteration.

mechanism is used to check for collisions with obstacles in the state space.

4. **Graph Insertion:** The new node and a new edge are inserted into the graph.
5. **Solution Check:** The graph is checked for a solution. If a solution has been found then we could stop. However, if an optimal path is required, the algorithm is usually run for a longer time for it to converge to an optimal path.

These steps lie at the heart of RRT. Iterating through these steps results in trees that are capable of solving motion planning problems in higher dimensions. Moreover, RRT could operate in non-holonomic environments with an acceptable time and space complexity. Also, with some modifications, RRTs are able to find optimal paths.

2.5. Voronoi Diagrams

An essential concept for the behaviour of RRTs is the Voronoi partitioning of a space. Given a set of points P the space is decomposed into regions V_i where for any point $y \in V_i$, $x \in P$ is the closest point to y using a predefined distance metric. Some Voronoi regions may have a greater Voronoi volume than other regions in the diagram. See Figure 1.

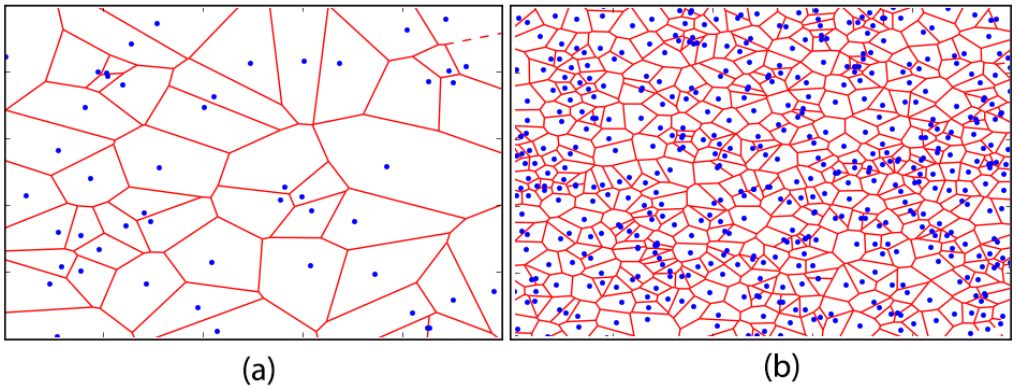

Figure 1. A Voronoi partitioning of the plane. (a) is partitioned by 50 uniformly sampled points, whereas (b) has 500 uniformly sampled points.

The Voronoi diagram will play an important role in the exploration of the trees as we will see in the coming sections.

2.6. Kinodynamic and Non-Holonomic Planning

Generally, there are two approaches to path planning problems. The kinematic approach concerns rigid bodies[5] that are subject to kinematic constraints. Kinematic constraints between rigid bodies inhibit the number of independent movements of those bodies[6] (Swaczyna, 2011), these could be joint limits of a (mechanical) system or obstacles. The constraints form restrictions on the motion of the rigid bodies. Typically, these restrictions come in two forms: holonomic constraints or non-holonomic constraints. (Non-)Holonomy is about the change of a point in the state space with respect to time, given the initial conditions. A kinematic constraint is said to be holonomic if it can be expressed in the following form: $f(q_i, t) = 0$, where q_i denotes the spatial coordinates and t denotes time. There are no other forces or derivatives with respect to time acting on the system. A system is said to be holonomic if all constraints in the system can be expressed in this form, in this case it is possible to revert back to the initial conditions. Consider a pendulum on a plane. The pendulum is a weight connected to a pivot through a rod (Figure 2). The pendulum can move in the x and y direction and with the length L of the rod held as a constant. We could write the constraint $x^2 + y^2 - L^2 = 0$, therefore this constraint is said to be holonomic; the contraints only depend on the spatial coordinates, it could go back to its initial conditions, i.e., the start position (configuration) of the pendulum. A kinematic constraint is said to be non-holonomic if it cannot be expressed in the holonomic form. The configuration of the system does not solely depend on its spatial coordinates. These non-holonomic constraints could arise from velocities of the rigid bodies but also from constraints on the state space. In the latter case, consider a rolling ball on a plane (Johnson, 2007). The ball may not slide or spin on the plane. The x and y coordinates denote its position, however these coordinates are not sufficient to describe its total configuration. Suppose that a red dot is painted on top of the ball, now let the ball roll in the x direction such that the red dot reappears on the top after rolling, do the same in the y direction, now roll the ball back to its starting position by rolling diagonally. The ball has now returned to the starting point but the red dot does not appear on top. Therefore, the x and y coordinates do not uniquely specify its configuration; we need to consider the orientation of the ball. This system is said to be non-holonomic (Figure 2). In short, the problems defined within non-holonomic systems are those in which mechanical or differential constraints must be considered.

The kinematic approach disregards forces that cause the inhibition of motion. In kinodynamic planning, the second approach, the rigid bodies are subject to the dynamics on the body, e.g., the effect of torques, velocities or acceleration on it (LaValle, 2006). The kinodynamic approach incorporates dynamic constraints. Generally, these are second order differential constraints. The kinodynamic approach is broader than the kinematic approach, in the sense that it also includes kinematic constraints and obstacle avoidance. The class of kinodynamic problems has been shown to be NP-hard (Donald et al., 1993) when three dimensions or more are considered. Algorithms that try to approximate solutions such as PRM have a hard time when the dimensions are increased and when non-holonomic constraints are considered. The roadmaps it constructs are mostly insufficient, it needs to control inputs to check whether the constraints have been satisfied. Nevertheless, there have been approaches using PRM to solve problems in non-holonomic systems (Karaman & Fraz-

[5] A rigid body is a solid body with little to no deformations when force is applied to it.
[6] The Degrees of Freedom

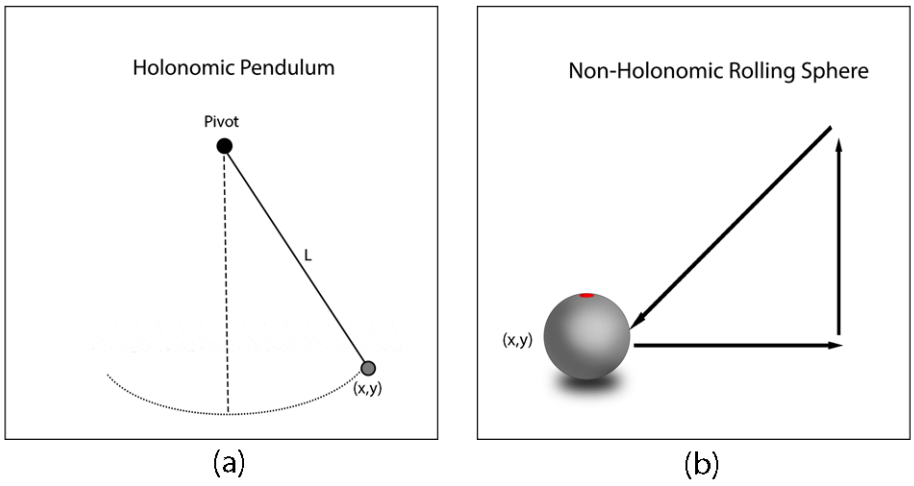

Figure 2. Frame (a) shows a pendulum on a plane, where the x and y coordinates denote its position and L is the length of the rod holding the weight. (a) is a holonomic system. Frame (b) shows a sphere on a plane. The configuration of the rolling ball does not solely depend on its spatial coordinates. This system is non-holonomic.

zoli, 2013; L. & Arenas, 2002; Overmars, 1998). The RRT's incremental and single query nature make RRT more robust with non-holonomic constraints. It was specifically designed to operate in non-holonomic environments (LaValle, 1998). As we previously described in Section 2.4, step two of the family of RDTs selects a node with a nearest neighbour procedure, hereafter in step three it is checked whether the defined (non-)holonomic constraints are satisfied. For example, consider Figure 3 where there exists a slight bias to the left. Control inputs are applied to the third step, yielding a slightly "bent" tree. RRT and its variants have been successfully applied in various environments with both holonomic and non-holonomic constraints such as Car-Like models, UAVs and robot-arms (Karaman et al., 2011a; Lan & Cairano, 2015; Adiyatov & Varol, 2013; Lee & Shim, 2014; Moon & Chung, 2015; Daniel et al., 2014)

In Section 3 we will fully elaborate on the growth of the tree, however in this paper we will only consider holonomic environments. This will lay the focus primarily on the behaviour of the algorithms and not on specific kinodynamic constraints. Moreover, we will only consider a two-dimensional grid.

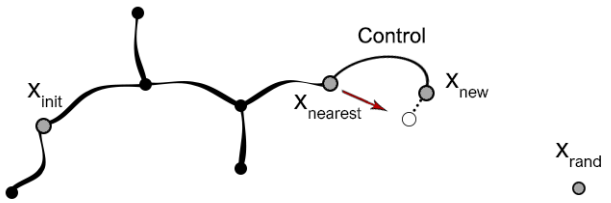

Figure 3. The control inputs are applied after a nearest neighbour is chosen for expansion. In the section on the RRT algorithm we will see that the Steer procedure returns a new node. If the system were to be (non-)holonomic this procedure would check if the new node satisfies these constraints.

3. Algorithms

This section is devoted to the algorithms of the RRT class, some of their variations and the properties they manifest. Three main variants of the RRT class are presented.

First, the basic RRT algorithm that was introduced in 1998 by M. LaValle and Kuffner is presented (LaValle, 1998). Subsequently, the properties of this algorithm are discussed along with some improvements that take advantage of these properties. Some of these properties are inherited by the second algorithm we will discuss: the RRT* algorithm introduced by Karaman and Frazzoli in 2010 (Karaman & Frazzoli, 2010). It is shown that this algorithm converges to an optimal solution almost surely, i.e., RRT* is asymptotically optimal. It is this quality of solution that the informed RRT* algorithm takes advantage of, making this the third algorithm we will discuss along with its enhancements on the regular RRT*.

The three mentioned algorithms lay the groundwork for many variations some of which will be discussed in this paper. The procedures and operations found in these algorithms are also found in their variant counterparts. However, their implementation and theory vary.

3.1. RRT

The RRT algorithm (Algorithm 1), was initially designed to have as few as possible arbitrary parameters for a consistent behaviour and ease of analysis. The incremental and sampling-based nature of the RRT algorithm makes it an attractive approach for non-holonomic and kinodynamic problems (LaValle, 2006). In this section we will first focus on the behaviour of the RRT algorithm, consequently establish some properties of RRT. We will then discuss its implicit Voronoi bias that gives rise to this behaviour and give an informal analysis of its convergence towards the sampling distribution. Subsequently, we will discuss its almost sure convergence towards a suboptimal solution. To finalise, we discuss some extensions, or small improvements, for path planning problems that may require some more effective approaches. These include, among others, bidirectional trees (RRT-Connect) (Kuffner &

LaValle, 2000), Goal-Bias-RRT (Sujith Kumar et al., 2010) and Re-configurable Random Forest (RRF) (Li & Shie, 2002). For the sake of clarity, we will always refer to the basic algorithm as just *RRT*.

3.1.1. The RRT Algorithm

Algorithm 1 RRT

Input: initial node x_{init}, max nodes $K \leftarrow \mathbb{N}$
$V \leftarrow \{x_{init}\}$; $E \leftarrow \emptyset$;
for $i = 0$ **to** K **do**
  $x_{rand} \leftarrow \text{Sample}(i)$
  $x_{nearest} \leftarrow \text{Nearest}(V, x_{rand})$
  $x_{new} \leftarrow \text{Steer}(x_{nearest}, x_{rand})$
  if $\text{ObstacleFree}(x_{nearest}, x_{new})$ **then**
    $V \leftarrow V \cup \{x_{new}\}$
    $E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$
  end if
end for
return (V, E)

The main idea of the RRT algorithm is to create a space-filling tree by incrementally biasing the search towards empty spaces and with each iteration effectively "push" the potential new nodes in the tree away from the nodes already added to V in previous iterations. Initially, a defined point in space (x_{init}) is added to V and the set of edges E is empty, as there is only one node explored. To extend the tree and create branches for a predefined maximum number of nodes K, at each iteration a point x_{rand} is sampled from a uniform distribution with the *Sample* function. This point is used to search for the nearest node in the V with respect to x_{rand}. Ultimately, a new node x_{new} is found by minimising the distance between x_{rand} and $x_{nearest}$. That is, x_{new} is the new node by performing one step with a distance smaller or equal to ϵ in the direction of x_{rand}. If the global constraints are satisfied[7], then the new (biased) node is added to V and a new edge from the nearest neighbour to the new node is added to E.

This process is shown in Figure 4, while Figure 5 shows a run of the RRT algorithm. The node x_{init} is in the upper-right corner, notice how the tree grows diagonally from left to right, effectively taking advantage of the open spaces. It is also important to mention that RRTs are not random walkers, i.e., an algorithm performing a random walk over X_{free} and choosing a node from V at random has a strong bias towards places already visited. The key difference is that RRT works by being biased towards empty places, since the selection of a node from V is based on a nearest neighbour function. This so called *Voronoi* bias will be further discussed in this section.

[7] Remember that we could add differential constraints or that obstacles could be interfering with the growth of the tree.

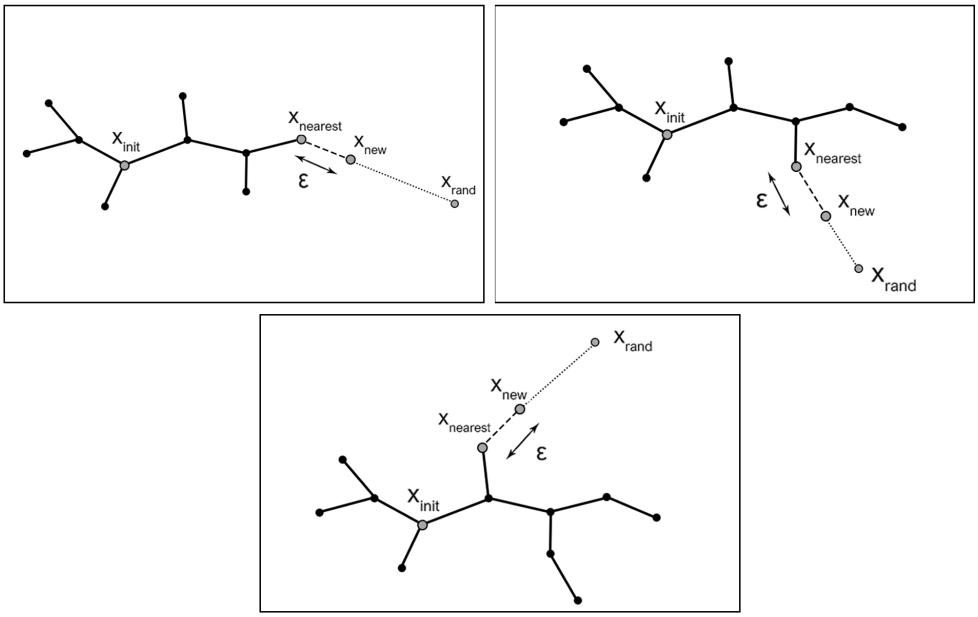

Figure 4. Three iterations of the RRT algorithm in chronological order.

Properties

What follows is a list of properties of RRT. Some of these properties make RRT attractive for a variety of motion planning problems.

1. The RRT algorithm requires few parameters and heuristics.
2. There is always a path between a node $v \in V$ and x_{init}, since it is necessary that $x_{init} \in V$ and each node has only one parent.
3. RRTs rapidly grow towards unexplored regions in X_{free}.
4. Under general assumptions, RRT is probabilistically complete. That is, the probability that a solution is found approaches one gradually.
5. The distribution of V converges towards the sampling distribution, essentially filling up the space with nodes. We discuss this briefly property in Section 3.1.3.
6. The probability that RRT converges to a suboptimal solution is one. This property will be proven in the subsection on Asymptotic Suboptimality (Section 3.1.4.

The sixth property tells us that RRT converges to a suboptimal path almost surely. However, because of the rapid exploration in the state space and the uniform sampling distribution, the paths in RRT come close to the true shortest paths. These properties, but especially the

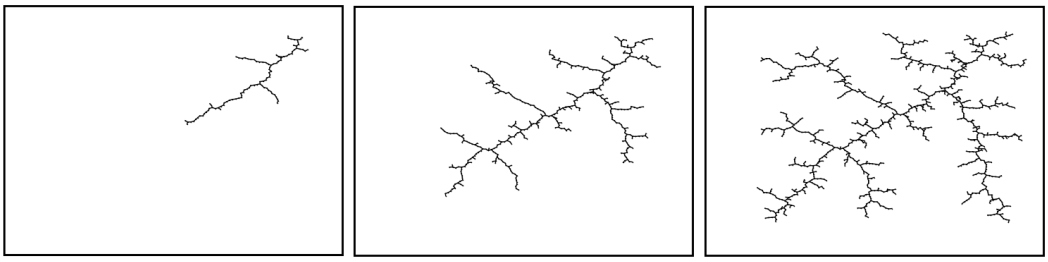

Figure 5. The generation of an RRT with $X_{free} = [0, 10] \times [0, 10]$, $x_{init} = (8, 9)$ and a stepsize $\epsilon = 0.1$. The third frame from the left is the completely generated tree with $K = 900$. In this generation no goal area was considered.

probabilistic completeness, is what leads to feasible planning and have been used in many applications (LaValle, 2006). These include a spacecraft in a simulated environment that is required to navigate through dangerous terrains, while at the same time considering gravity (LaValle & Kuffner, 2000).

3.1.2. Bias towards Unexplored Regions

We will give an intuition for the behaviour of the RRT algorithm with respect to its incremental growth in X_{free}. This intuition is explained by considering the Voronoi diagram of a set of points in a plane. In this case, the RRT is conceptualised using the nodes in V as the set of points partitioning the plane. This way, each node has a Voronoi region with its corresponding Voronoi volume.

The set V is constructed by taking random samples and computing the nearest neighbour for the random simple from V. As mentioned earlier, this leads to an incremental construction of a search tree biased towards open spaces. This is in essence the inherent Voronoi bias of RRTs and causes the search tree to "rapidly" explore its search space. Note that the Voronoi diagram of the search tree is never explicitly computed in (Algorithm 1). Although, some implementations exist where the Voronoi bias is fully taken advantage of by constructing the Voronoi diagram of the search tree at each iteration (Lindemann & LaValle, 2004). Using the diagram one could easily decide which portions of the search space need to be explored. Consider Figure 6 where a tree with four nodes partitions X_{free} such that the volume of area V_4 is greater than V_1, V_2, V_3 and V_5 where V_i[8] is a Voronoi region. Since the nearest neighbour procedure will return the nearest node to the (uniformly) sampled point, the probability that the sampled point will fall in the V_4 region is greater than that it will fall in the other regions. Such that the probability that node x_4, which corresponds to V_4, is chosen for expansion will be greater than the probability that some other node will be chosen as the nearest neighbour. More generally, the probability that a node $x_i \in V$ is chosen for expansion is proportional to the volume of V_i. This is the reason for the bias of the search tree towards open spaces.

[8]Not to be confused with V that denotes the set of nodes of the tree

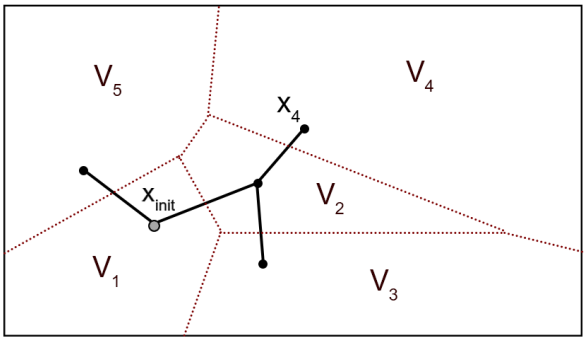

Figure 6. The Voronoi partition of the search space given a tree that consists of five nodes. The Voronoi region V_4 has the largest volume.

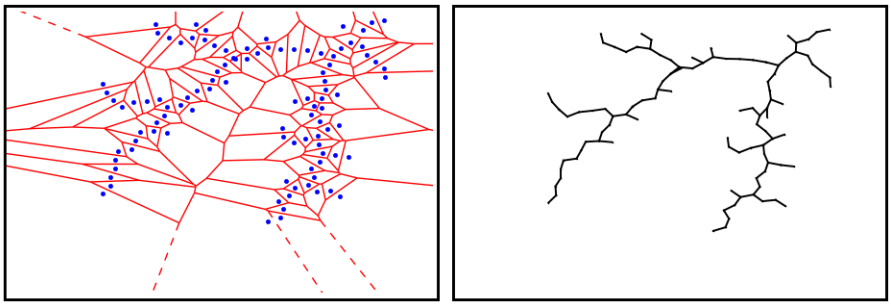

Figure 7. The left frame shows the Voronoi partition of the search space given a tree that consists of 100 nodes starting at the upper right corner, while the right frame shows this tree. The blue dots in the Voronoi partition are the nodes of the tree.

3.1.3. Convergence towards the sampling Distribution

In this section the convergence towards the sampling distribution is discussed briefly and informally. The reader is invited to read (Kuffner & LaValle, 2000) for a fully formal elaboration.

RRT incrementally adds a node to the tree (set of vertices V) via a uniform sampling distribution. As the size of V increases, the uncovered space of the graph gets filled, since a new node is added within ϵ from the chosen nearest-neighbour. This way, the added nodes will follow the same probability density of the state space because the volume of the uncovered space approaches zero as the uncovered space gets gradually covered.

3.1.4. asymptotic Suboptimality

In this subsection we will discuss the probabilistic suboptimality of RRT by (briefly) providing the proof, as an allusion to the probabilistic optimality of RRT*, which will be introduced later on. For an in-depth proof see (Karaman & Frazzoli, 2010)

$\{C_i^{RRT}\}_{i\in\mathbb{N}}$ is a sequence of random variables each denoting the cost of a minimum-cost path in the tree (RRT) at iteration i.

First, we need to see that the limit of C_i^{RRT} exists: $C_{i+1}^{RRT}(\omega) \leq C_i^{RRT}(\omega)$ for all $\omega \in \Omega$ and all $i \in \mathbb{N}$, since the cost of a path can only increase if a new node is added to the tree. Furthermore, $C_i^{RRT}(\omega) \geq c^*$ holds for all $\omega \in \Omega$ and all $i \in \mathbb{N}$ because the minimum-cost path cannot be more optimal than c^*. Therefore $\{C_i^{RRT}\}_{i\in\mathbb{N}}$ is non-increasing and lower-bounded by c^*. Hence:

$$\lim_{i\to\infty} C_i^{RRT}(\omega) \text{ for all } \omega \in \Omega. \tag{1}$$

In order to prove the almost sure suboptimality of RRT three assumptions must be made:

Assumption 1: The set of all points on the optimal path has a volume of zero[9]. With this assumption it is not possible to sample an optimal path, since the probability of sampling a point on this path is almost surely zero.

Assumption 2: The sampling function cannot be constructed such that it generates an optimal path.

Assumption 3: for all $\sigma_1, \sigma_2 \in \Sigma_{X_{free}}$ it must hold that $c(\sigma_1) \leq c(\sigma_1 \mid \sigma_2)$. That is, concatenating two paths yields a path that cannot have a lower cost.

Since we know from equation 1 that the limit exists and given the three assumptions, we can now conclude that

$$\mathbb{P}\left\{ \lim_{i\to\infty} C_i^{RRT} > c^* \right\} = 1 \tag{2}$$

That is, the probability of the cost of a minimum-cost path being greater than the cost of the optimal path is one as i grows to infinity. In other words, as new nodes are (infinitely) added to the tree, the probability of a path in the tree being the optimal path becomes zero.

3.1.5. Problems

Although the RRT algorithm is easy to implement, rapidly explores its space and is capable of handling high dimensional environments, it does not come without its problems. RRT is suboptimal: the generated tree does not improve on its found solution. The space is constrained and RRT usually gets stuck with a first found solution. Moreover, the state

[9] More precisely, a Lebesgue measure of zero.

space is explored uniformly, this could lead to a slow (or premature) convergence towards the goal. Furthermore, the set containing the nodes increases with each iteration, which in turn means that the search for the nearest neighbour becomes increasingly more costly. Several approaches have been proposed, such as the use of a different data structure: a *kd*-tree[10] (Bentley, 1975), as opposed to a naive implementation. Good results have also been obtained using an approximate nearest neighbour search (Arya et al., 1999; Fu & Cai, 2016; Karaman & Frazzoli, 2010) instead of an exact approach. Furthermore, RRT has difficulty with planning problems that look like "bug traps" (Figure 8). In the subsection on the variants of RRT, we will discuss an approach that helps to solve this problem: Bidirectional RRT.

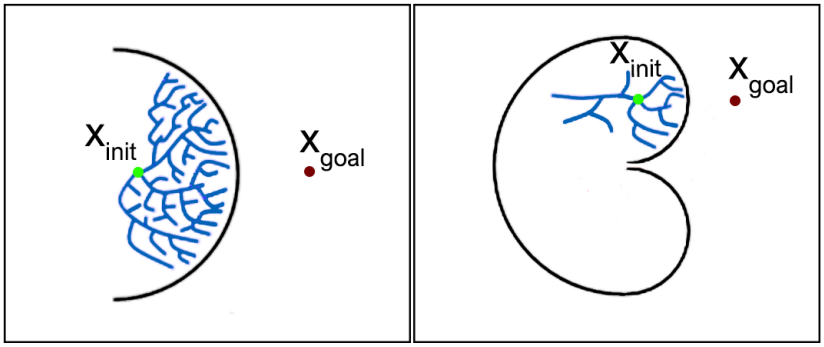

Figure 8. These two problems are difficult for RRT to solve, especially the so called bug trap (right frame).

RRT Variants

In this subsection some variants to the RRT algorithm will be briefly discussed. Some of these are simple alterations to the base algorithm.

3.1.6. Biased-RRTs

A way to improve the convergence rate towards the goal area is by biasing the growth of the tree. A weakness of RRT is that it does not consider the path costs. In (Urmson & Simmons, 2003) a heuristic quality function is used to bias the growth of the tree towards the goal by incorporating a preference for shorter paths. Another approach to bias the growth towards the goal area is to sample a point from the goal area at each iteration with some predetermined probability. In both of these improvements, if too much bias is added, the tree may get quickly trapped in local minima. A gradual bias towards the goal is preferred. The gradual bias is usually based on the distance from the goal to the nearest node in the tree (Sujith Kumar et al., 2010). An additional improvement to bias the RRT even further, but this time toward the random node x_{rand}, is to keep adding nodes to the tree until x_{rand}

[10] A *kd*-tree is a datastructure forming a binary tree with constraints to organise k-dimensional points.

has been reached or an obstacle impedes the growth. The idea is to extend the tree with many "steps", instead of just one step of length ϵ (Figure 9).

3.1.7. RRT-Connect and Reconfigurable Random Forests

A solution to the bug-trap problem is to build two RRTs, one originating at the start and another originating at the goal. One RRT will add x_{new} to the graph, while the other RRT will try to grow towards x_{new}, the trees are connected when it is possible to reach x_{new} directly. The main idea is thus to grow these two trees and connect them whenever they can "see" each other, i.e., when no obstacles limit their view (Kuffner & LaValle, 2000). Bi-directional search has shown to be more efficient, especially in three-dimensional planning problems (Li & Shie, 2002). In (Li & Shie, 2002) an extension of the RRT-Connect is considered, where not two but multiple trees are created and connected with each other to create a forest. The authors use a data structure named Reconfigurable Random Forests, the trees are pruned and invalid nodes are discarded. This way the forest is kept small and it is shown that it is robust with environmental changes: the trees can be reconfigured when an obstacle slices through a connection.

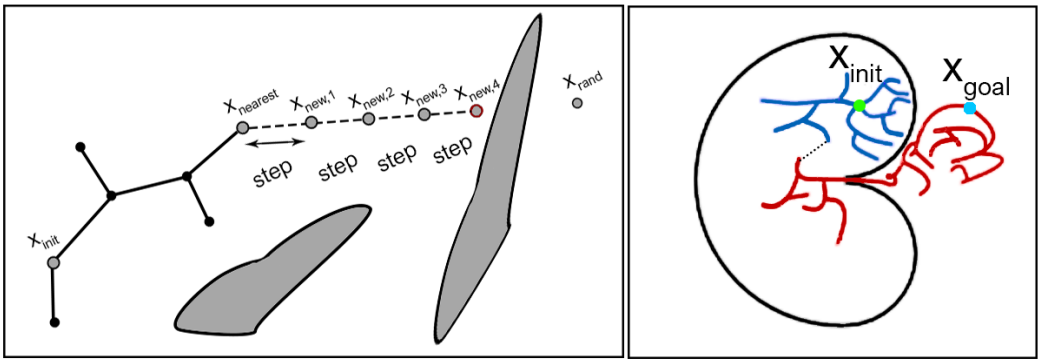

Figure 9. In the left frame whenever the nearest neighbour has been selected, grow towards x_{rand} until it is reached. In the right frame the bug trap problem becomes easier when a multi-directional tree is used.

Although these algorithms increase the convergence rate or improve the growth of the tree, they keep the suboptimal property of RRT.

3.1.8. Conclusion

RRT works by incrementally growing a tree, its implicit Voronoi bias makes the tree rapidly explore open spaces. The algorithm is sampling-based and has the property of probabilistic completeness, however the base algorithm does not use the path cost to improve the found solution. This drawback results in asymptotic suboptimal behaviour. It will almost surely find a suboptimal path. Some variants exist that can guide the tree or that increase the probability of finding the goal but RRT will maintain a suboptimal path.

3.2. RRT*

The RRT* algorithm (Algorithm 2), operates similarly to the RRT algorithm, in the sense that it searches the state space incrementally with a bias for open spaces. RRT* inherits the first five properties we discussed in the section on RRT. However, the RRT* algorithm has the favourable property of an almost sure convergence to an optimal solution (Karaman & Frazzoli, 2010). Similar to the section on RRT, in this section we will focus on the behaviour of the RRT* algorithm and discuss its improvements on the RRT algorithm. One of which is the almost sure convergence to an optimal solution as the number of samples approaches infinity. We will finalise by regarding some problems and, in the section on the RRT* variants, some refinements on this algorithm will be discussed. These include RRT*-FN (Adiyatov & Varol, 2013) and RRT*-Smart (Nasir et al., 2013).

Although several algorithms have been proposed to guide the growth of an RRT towards finding an optimal path using heuristics (Urmson & Simmons, 2003) or simply by exploiting the Voronoi diagram of the search, the quality of the found solution falls short, especially when an optimal path to the goal is needed. (see the section on RRT's suboptimal convergence). The problem is mainly due to RRT's lack of consideration for the path costs. In (Urmson & Simmons, 2003) the growth of an RRT is guided by considering the lowest path cost to a node and using a quality measure defined for each node. Instead of computing only the nearest neighbour from V, the k-nearest neighbours are considered. Although the path costs are considered, and although the guidance gives better results, this RRT variant retains its inherent probabilistic suboptimality.

In order to improve the rate of convergence of the RRT algorithm, RRT* was proposed (Karaman & Frazzoli, 2010). RRT* is based on the RRG[11] algorithm (Karaman & Frazzoli, 2009), which is a graph-based[12] algorithm with the property of asymptotic optimality that RRT* inherits. However, the cost of a path is taken into account to choose the node with the lowest cost from x_{init} from the k-nearest neighbours defined by a ball with radius r. Additionally, to make sure that the minimal cost is maintained between the nodes, a rewiring operation restructures the tree by removing and adding edges from the tree.

3.2.1. The RRT* Algorithm

As was previously briefly mentioned, the RRT* algorithm introduces two important alterations to the RRT algorithm. Both of which use the cost of the path from x_{init} to the node in question. RRT* generates its tree in a similar way as RRT; at each iteration, x_{rand}, $x_{nearest}$ and x_{new} are assigned values in the same manner it has been discussed for RRT. The RRT* algorithm, however, has three new procedures that give rise to its optimal behaviour, readers are advised to read (Karaman & Frazzoli, 2010) for more details.

- The *Near* function returns a set nodes $V' \subseteq V$ of the nearest neighbours in an r-ball $B_{r,x}$ with radius r centered at x. To guarantee an almost-sure convergence to an optimal

[11] Rapidly-Exploring Random Graph.

[12] The nodes of a tree-based algorithms have only one parent, whereas the nodes in a graph-based algorithms could have multiple parents; cycles are possible.

solution, the *Near* function computes the radius around x according to:

$$r_i = \min\left\{\left(\frac{\gamma}{\zeta_d}\frac{\log(n)}{n}\right)^{1/d}, \eta\right\}, \tag{3}$$

Where η is a predefined maximum radius, n is the length of V, ζ_d is the volume of $B_{r,x}$, $\gamma > 0$ a constant, such that

$$\gamma \geq 2^d(1+\frac{1}{d})\mu(X_{free}), \tag{4}$$

where μ denotes the volume of the free space.
However, it is possible to use a predefined constant radius r_i but at the cost of the asymptotic optimality[13].

- The *Parent* function chooses the parent from the set of neighbours, returned by the *Near* function, of a new node x_{new} where the chosen parent has the lowest cost to reach x_{new}.
- The *Rewire* procedure deletes an edge from the graph between two nodes x, y if a path with a lower cost is found for another node $z \in B_{r,x}$ and $z \in V$, that goes through x. A new edge between x and z to the graph is consequently added to the graph.

Algorithm 2 RRT*

```
Input: initial node x_init, max nodes K ← ℕ
V ← {x_init}; E ← ∅;
for i = 0 to K do
    x_rand ← Sample(i)
    x_nearest ← Nearest(V, x_rand)
    x_new ← Steer(x_nearest, x_rand)
    V ← V ∪ {x_new}
    if ObstacleFree(x_nearest,x_new) then
        X_near ← Near(V, x_new)
        x_nearest ← Parent(X_near, x_nearest, x_new)
        E ← E ∪ {(x_nearest, x_new)}
        E ← Rewire(X_near, E, x_new)
    end if
end for
return (V, E)
```

When the new point x_{new} has been generated and added to the tree, the Near function uses x_{new} and V to get the nearest neighbours. Subsequently, from this set of nearest neighbours a parent is chosen by looking at the node x_{near} with the lowest path cost to x_{new}. This process is schematically shown in Figure 11. If there are no nodes within $B_{r_i,x_{rand}}$ then $x_{nearest}$ is used as the solitary nearest neighbour, similar to RRT.

Subsequently, the tree is rewired to find a potentially better path through x_{new} by examining the current cost of the path to x_{near} and comparing it to the cost of the path to

[13] Although, in practice, there does not seem to be much difference.

Algorithm 3 Parent

Input: X_{near}, $x_{nearest}$, x_{new}
for $x_{near} \in X_{near}$ **do**
  if ObstacleFree(x_{near},x_{near}) **then**
    $c' \leftarrow x_{near}$.cost + c(Line(x_{near}, x_{new}))
    if $c' < x_{new}$.cost **then**
      $x_{nearest} \leftarrow x_{near}$
    end if
  end if
end for
return $x_{nearest}$

Algorithm 4 Rewire

Input: X_{near}, E, x_{new}
for $x_{near} \in X_{near}$ **do**
  if ObstacleFree(x_{new},x_{near}) **and** x_{near}.cost > x_{new}.cost + c(Line(x_{near}, x_{new})) **then**
    $x_{parent} \leftarrow x_{near}$.parent
    $E \leftarrow E \setminus \{(x_{parent}, x_{near})\}$
    $E \leftarrow E \cup \{(x_{new}, x_{near})\}$
  end if
end for
return E

x_{near}, in the case it would go through x_{new}. The rewiring process is shown in Figure 12. A sample trace of RRT* is shown in Figure 10. As the number of nodes added to the tree increases, the Parent and Rewiring functions incrementally improve the path cost of RRT* yielding asymptotic optimality.

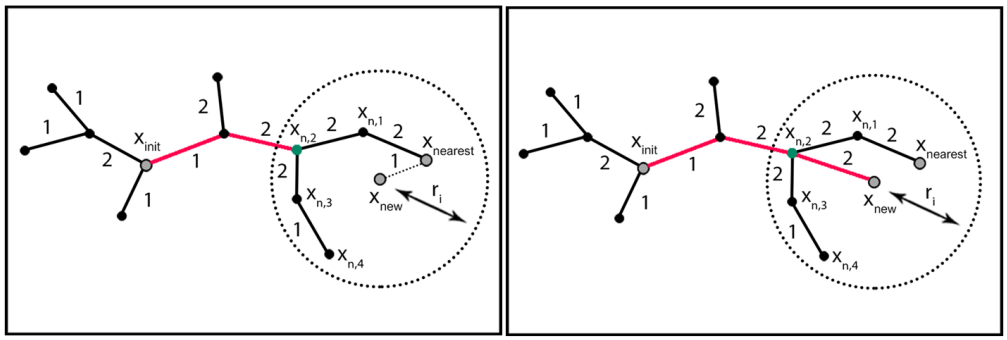

Figure 11. The (Choose) Parent procedure in the RRT* algorithm is shown. In the left frame $x_{nearest}$ is chosen as the parent of x_{new}. The path costs to each $x_{n,i}$ within $B_{r,x}$ is examined. The path up to node $x_{n,2}$ has the lowest cost. Therefore, $x_{n,2}$ will be chosen as the parent.

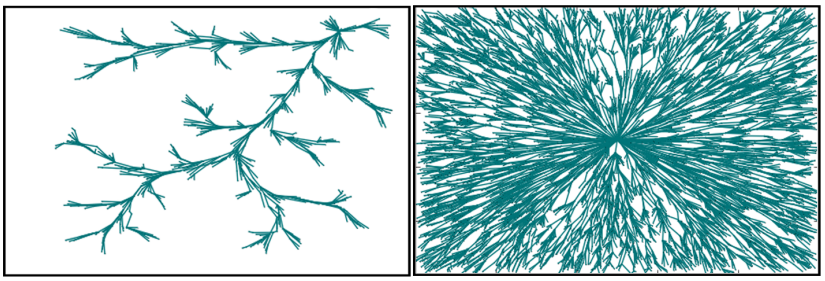

Figure 10. A run of RRT* with $X_{free} = [0, 10] \times [0, 10]$ with a stepsize $\epsilon = 0.1$, for the left frame $x_{init} = (8, 9)$ and $K = 900$; for the right frame $K = 8000$ and $x_{init} = (5, 5)$. In this generation no goal area was considered.

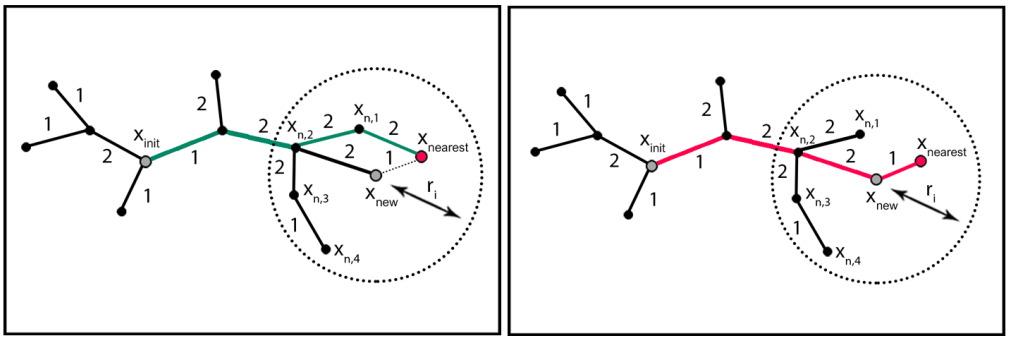

Figure 12. The Rewire procedure in the RRT* algorithm is shown. In the left frame the path highlighted in green has a total path cost of seven from x_{init} to $x_{nearest}$. The Rewire procedures checks if it would be cheaper if an edge went from x_{new} to $x_{nearest}$ yielding the path highlighted in red in the right frame. The two paths are compared; the path running through x_{new} yields a cost of six, which is cheaper than the green path. The edge between $x_{n,1}$ and $x_{nearest}$ is deleted and a new edge $(x_{new}, x_{nearest})$ is added to E. The rewire procedure computes this process for each $x_{near} \in X_{near}$

3.2.2. Properties

What follows is a summary of properties of RRT*, In addition to the first five properties discussed for RRT.

1. The probability that RRT* converges to an optimal solution is one. This property will be proven in the subsection on Probabilistic Optimality.
2. RRT* has a similar time complexity to RRT; the processing of the tree in both algorithms takes $\mathcal{O}(n \log n)$ and the querying part for both algorithms is $\mathcal{O}(n)$. Note that

we will not discuss time complexity in this paper, see (Karaman & Frazzoli, 2010) and (Karaman & Frazzoli, 2011) for more details.

To demonstrate the first property of RRT* consider (Professor Emilio Frazzoli et al., 2009), where an autonomous forklift was used RRT to park in front a truck while it evaded obstacles. An obstacle placed near the track made the forklift loop around the truck to find a suitable parking spot. The RRT found in this planning problem a suboptimal path. RRT* was used to overcome this problem and in turn fixed the loop RRT created to find a shorter path to park the forklift. The jagged and chaotic paths created by RRT were fixed.

3.2.3. Asymptotic Optimality

In this subsection we will discuss the asymptotic optimality of RRT* by providing a proof for this assertion.

Let $\{C_i^{RRT^*}\}_{i\in\mathbb{N}}$ be a sequence of random variables each denoting the cost of a minimum-cost path in the tree (RRT*) at iteration i

Note that, in the same manner, as explained in the RRT subsection on probabilistic suboptimality, the limits for $C_i^{RRT^*}$ and $\{C_i^{RRT^*}\}_{i\in\mathbb{N}}$ exist.

In addition to assumptions 1, 2 and 3 from the subsection on asymptiotic suboptimality of RRT, three new assumption must be made to prove the optimality of RRT*:

Assumption 4: For all $\sigma_1, \sigma_2 \in \Sigma_{X_{free}}$ it must hold that $c(\sigma_1 \mid \sigma_2) = c(\sigma_1) + c(\sigma_2)$.

Assumption 5: Any two paths that are close to each other must have similar costs.

Assumption 6: There must be sufficient space surrounding the optimal paths. This way the free space will allow convergence.

Given that the (choose) *Parent* function always chooses the node within the r-ball $B_{r,x_{new}}$ on a path with the lowest cost, given that the tree is rewired to find shorter paths within the r-ball and that assumptions 4, 5 and 6 hold, then RRT* converges to c^* almost-surely:

$$\mathbb{P}\left\{\lim_{i\to\infty} C_i^{RRT^*} = c^*\right\} = 1 \tag{5}$$

This concludes the proof for path optimality in RRT*.

3.2.4. Problems

RRT* tries to find the optimal path between the start and the goal in the free space but it does this by exploring the whole region in the planning domain, long after the goal area has been found. This results in a slow convergence rate. The algorithm generally needs an infinite amount of time to find an optimal path (Karaman & Frazzoli, 2011). Also, due to the vast exploration of the whole space, the algorithm suffered from memory issues as the tree became larger. (Adiyatov & Varol, 2013).

RRT* Variants

In this section we will briefly discuss two RRT* variants that try to improve on the shortcomings of RRT*: RRT*-FN and RRT*-Smart.

3.2.5. RRT*-FN

RRT*-FN (RRT* Fixed Nodes) (Adiyatov & Varol, 2013) imposes a restriction on the number of nodes expanded by RRT* to handle the memory issues that RRT* experiences. RRT*-FN uses two new procedures: a global and local node removal. Local node removes nodes with only a single child when the tree is rewired. When the limit of nodes has been reached, a new node can only be added to tree if a node is removed somewhere else. This way RRT*-FN lowers the memory consumption of RRT* and does not affect the convergence rate of RRT* significantly.

3.2.6. RRT*-Smart

In (Nasir et al., 2013) a new version of RRT* is proposed to ensure a faster convergence of the algorithm. RRT*-Smart is an informed version of RRT* and improves significantly on the slow convergence problem of RRT*. RRT*-Smart employs three phases. In the first phase the regular RRT* is used to find the goal. In the second phase the path is optimised. The optimisation of the path yields biasing points named beacons for the so called intelligent sampling. The third phase uses these beacons from phase two to intelligently draw samples near the optimised path.

After the goal node has been found, the path to the goal is optimised in the second phase. It is checked if there is a shorter, more direct, path between two nodes. This path optimisation technique uses triangular inequality (Figure 13) to connect two points, x and y, diagonally when a third point z is initially the point connecting x and y. The path between x and y is shorter than the path through z by the triangular inequality. Prior to adding the edge to the graph, it is checked whether the path between x and y is a viable one. This method is applied to all the nodes in the path and, if the optimisation finds a better route it reduces the number of nodes in that path. The nodes in the path are used in the intelligent sampling procedure.

The intelligent sampling procedures uses the nodes in the path as beacons. The beacons serve as the centre of a ball radius. Intelligent sampling was proposed, since the path optimisation had difficulty solving problems with complex shapes of obstacles (circular, polygonal etc). Therefore, the tree is biased to draw more samples along the beacons (the optimised path) to smooth out the path even further towards the goal. It uses a biasing radius that serves as the radius of the ball centred at the beacon. The radius is chosen according to the requirements of the planning problem.

The optimisation of the path together with the intelligent sampling procedure ensures a significant improvement in the convergence rate of RRT*; it needs fewer iterations to reach an optimal solution and a an accelerated rate. Yet, RRT-Smart has the same space and time complexity of RRT*. Furthermore, the biasing radius introduces a trade-off between the exploration rate of the algorithm and the convergence rate. This biasing radius is not

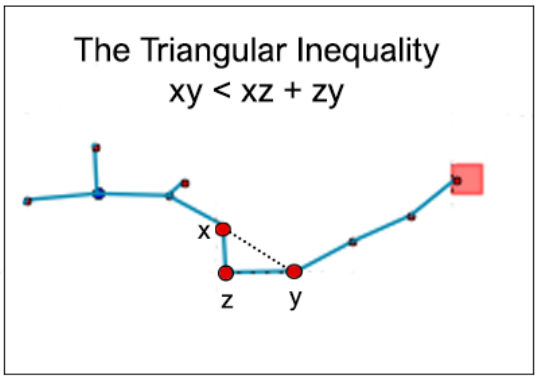

Figure 13. The path optimisation uses the triangular inequality to smooth the path from the start to the goal in RRT*.

automated it needs fine tuning according to the problem at hand. By introducing the beacons for further bias it seem to be violating the assumption of uniform density of the RDT family.

3.2.7. Conclusion

RRT* grows a tree incrementally, much in the same way RRT does. It added two added procedures that look at the path cost. The Parent procedure chooses the parent from a ball radius for a new node and the rewire procedure restructures the tree to find less costly paths considering the new node. With these new procedures and under general assumptions RRT* has been proven to be asymptotically optimal. However, the rate of convergence is slow because RRT* continues to explore the whole state space. RRT*-Smart is a variant that tries to improve on the convergence speed by finding shorter paths using the triangular inequality and intelligent sampling, where some nodes are used as beacons and the growth is further biased towards the goal. RRT-FN* solves the increased memory consumption by limited a fixed number of nodes.

3.3. Informed RRT*

The Informed RRT* algorithm (Gammell et al., 2014b) is displayed in Algorithm 5. This algorithm tackles the problem of slow convergence in RRT* by focusing the search on the goal area after it has been found by the regular RRT* algorithm. This algorithm inherits the optimal properties of RRT*, since the core of the base algorithm remains untouched, while demonstrating an order-of-magnitude improvement and in some cases[14] a linear convergence to the solution. We will focus on the behaviour of the informed RRT* algorithm and discuss its improvements on the RRT* algorithm. We will end this section with some disadvantages and we will briefly mention four algorithms that proceed to improve on the

[14] e.g., in the cases that there are no obstacles inhibiting the growth of the tree.

shortcomings of the algorithm.

Several variants exist to enhance the rate of convergence in RRT*. As seen in earlier sections, the RRT*-Smart algorithm makes use of a path optimization technique and intelligent sampling to confront the slow convergence. RRT$^{\#}$ (Arslan & Tsiotras, 2013) addresses the problem with two added procedures: exploration and exploitation. While exploration receives information about the search space, exploitation keeps track of the most promising nodes using the information acquired by the exploration procedure. Furthermore, in (Karaman et al., 2011b) two new key procedures were added to the RRT* algorithm, committed trajectories and branch-and-bound tree adaptation to make more efficient use of computation time. A heuristic estimate of the cost is used to remove nodes from the tree. In (Akgun & Stilman, 2011; Jordan & Perez, 2013) bidirectional trees (RRT*) are used with (multiple) heuristics, the algorithms demonstrate a better rate of convergence.

Informed RRT* tries to minimise the cost of the path by directly taking the samples in the region between the start and the goal bounded by an n-dimensional ellipsoid[15]. In this manner Informed RRT* limits the solutions only to the paths within the ellipsoid, i.e., the heuristic sampling domain, and not the whole free space, improving the convergence rate and increasing the probability of finding difficult paths. It is a simple extension of the RRT* algorithm, it makes no additional assumptions and, unlike other algorithms that use heuristic biasing, it does not expand nodes that cannot improve the solution.

3.3.1. The Informed RRT* Algorithm

In Algorithm 5 the Informed RRT* procedure is shown. The differences with RRT* are shown in green. Informed RRT* works similarly to the RRT* algorithm, the key difference being that after a path is found the algorithm narrows down the search space by only focusing on the space between the start and the goal, while gradually shrinking the ellipsoid to find the path with the smallest cost.

Informed RRT* works with a so called admissible heuristic. An admissible heuristic is one that never overestimates the real cost of a path, which generally means that for all $\mathbf{x} \in X$[16], $f(\mathbf{x}) \leq f^*(\mathbf{x}) = c(\sigma*)$, where $f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x})$. The cost of the path from $\mathbf{x}_{init}$ to $\mathbf{x}$ is denoted as $g(\mathbf{x})$ and $h(\mathbf{x})$ denotes the estimate from $\mathbf{x}$ to the goal. Since $f^*(\mathbf{x})$ is usually not known, $f(\mathbf{x})$ tries to estimate this function using an admissible heuristic[17] by increasing the probability of sampling from a subspace that can only lead to an improvement of the current path cost to the goal. Let $X_f \subseteq X$[18] be this informed subspace. The informed subspace is a hyperellipsoid defined as

$$X_f = \{\mathbf{x} \in X | \|\mathbf{x}_{start} - \mathbf{x}\| + \|\mathbf{x} - \mathbf{x}_{goal}\| \leq c_{best}\}, \tag{6}$$

With c_{best} being the cost of the current solution The subspace is schematically shown in Figure 14.

[15] More precisely, a prolate hyperellipsoid, i.e., an elongated symmetrical ellipsoid.

[16] Note that in this section we work with a slightly different notation. The points in the state space are vectors

[17] The Euclidean distance is a known admissible heuristic

[18] X_f not to be confused with X_{free}

A point $\mathbf{x}_f$ is uniformly sampled from X_f by first sampling $\mathbf{x}_{ball}$ from a unit n-ball and then computing the following procedure:

$$\mathbf{x}_f = \mathbf{C}\mathbf{L}\mathbf{x}_{ball} + \mathbf{x}_{centre}, \tag{7}$$

- Where $\mathbf{C}$ performs a rotation and is defined by:

$$\mathbf{C} = \mathbf{U}\text{diag}\{1, ..., 1, \det(\mathbf{U})\det(\mathbf{V})\}\mathbf{V}^T, \tag{8}$$

  where $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{d\times d}$ are unitary matrices and can be obtained with $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{M}$ via singular value decomposition (SVD). furthermore $\mathbf{M}$ is defined as,

$$\mathbf{M} = \mathbf{a_1}\mathbf{1}_1^T, \tag{9}$$

  here $\mathbf{1}_1^T$ denotes the first column of the identity matrix and $\mathbf{a}_1 = (\mathbf{x}_{goal} - \mathbf{x}_{init})/\|\mathbf{x}_{goal} - \mathbf{x}_{init}\|$.

- Where $\mathbf{L}$ performs a transformation and is defined as,

$$\mathbf{L} = \text{diag}\left\{\frac{c_{best}}{2}, \frac{\sqrt{c_{best}^2 - c_{min}^2}}{2}, ..., \frac{\sqrt{c_{best}^2 - c_{min}^2}}{2}\right\}, \tag{10}$$

  c_{min} here denotes the real distance between $\mathbf{x}_{init}$ and $\mathbf{x}_{goal}$, often defined by the Euclidean distance.

- Where $\mathbf{x}_{centre}$ is the centre of the ellipsoid translating the ellipsoid and is defined by,

$$\mathbf{x}_{centre} = \frac{(\mathbf{x}_{goal} + \mathbf{x}_{init})}{2}. \tag{11}$$

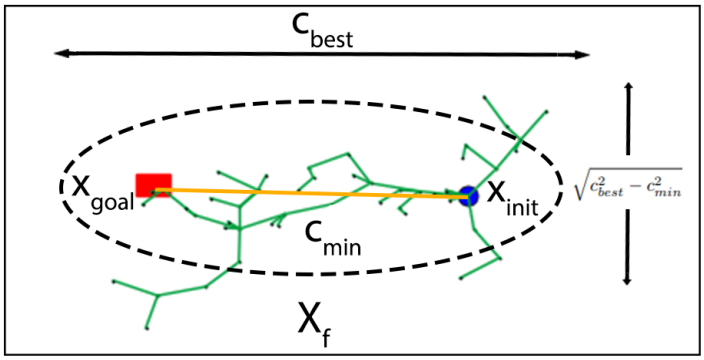

Figure 14. A view of the ellipsoid computed by Informed RRT*. c_{min} is the real distance between $\mathbf{x}_{goal}$ and $\mathbf{x}_{init}$. c_{best} is the cost of the best path found thus far. The ellipsoid shrinkes as c_{best} approaches the real distance. After the goal has been found, all samples are taken from within this ellipsoidal area.

Hence, the new procedure added by the Informed RRT* algorithm is the *Sample* procedure. This procedure draws, uniformly, samples from the ellipsoid with the procedure as seen in

equation (7). The rotation matrix $\mathbf{C}$ needs to be computed only once, before iterating. In Algorithm 5 line numbers 5-11 show the computation of $\mathbf{C}$ as seen in equations (7), (8) and (9). At the start of every iteration the theoretical (real) distance between the start and the goal are obtained. Note that c_{min} will only be useful when the goal has been found by RRT*, i.e., if $\mathbf{x}_{new}$ is contained in the goal region. The Sample procedure (Algorithm 6) draws samples in the usual manner if a goal has not been found yet. However, when the goal has been found and the cost of the current solution is computed, then Sample procedure computes $\mathbf{x}_{ball}$, $\mathbf{x}_{centre}$ and the transformation $\mathbf{L}$ (equation 10) needed to obtain $\mathbf{x}_f$. In Figure 15 a sample trace of the algorithm is shown. The reader is encouraged to read

Algorithm 5 Informed RRT*

1: **Input:** initial node x_{init}, max nodes $K \leftarrow \mathbb{N}$
2: $V \leftarrow \{x_{init}\}$; $E \leftarrow \emptyset$;
3: $X_{soln} \leftarrow \emptyset$
4: $c_{best} \leftarrow \infty$
5: $\mathbf{x}_{centre} \leftarrow (\mathbf{x}_{init} + \mathbf{x}_{goal})/2$
6: $c_{min} \leftarrow$Line$(\mathbf{x}_{init}, \mathbf{x}_{centre})$
7: $\mathbf{a}_1 \leftarrow (\mathbf{x}_{goal} - \mathbf{x}_{init})/\|\mathbf{x}_{goal} - \mathbf{x}_{init}\|$
8: $\mathbf{id}_1 \leftarrow \mathbf{1}_1^T$
9: $\mathbf{M} \leftarrow \mathbf{a}_1\mathbf{id}_1$
10: $\mathbf{U\Sigma V}^T \leftarrow$ SVD$(\mathbf{M})$
11: $\mathbf{C} \leftarrow \mathbf{U}$diag$\{1, ..., 1,det(\mathbf{U})det(\mathbf{V}\}\mathbf{V}^T$
12: **for** $i = 0$ **to** K **do**
13: $c_{min} \leftarrow \min_{x_{soln} \in X_{soln}}\{\text{cost}(x_{soln})\}$
14: $\mathbf{x}_{rand} \leftarrow$ Sample$(c_{best}, c_{min}, \mathbf{x}_{centre}, \mathbf{C})$
15: $\mathbf{x}_{nearest} \leftarrow$ Nearest$(V, \mathbf{x}_{rand})$
16: $\mathbf{x}_{new} \leftarrow$ Steer$(\mathbf{x}_{nearest}, \mathbf{x}_{rand})$
17: $V \leftarrow V \cup \{\mathbf{x}_{new}\}$
18: **if** ObstacleFree$(x_{nearest}$,$\mathbf{x}_{new})$ **then**
19: $X_{near} \leftarrow$ Near$(V, \mathbf{x}_{new})$
20: $\mathbf{x}_{nearest} \leftarrow$ Parent$(X_{near}, \mathbf{x}_{nearest}, \mathbf{x}_{new})$
21: $E \leftarrow E \cup \{(\mathbf{x}_{nearest}, \mathbf{x}_{new})\}$
22: $E \leftarrow$ Rewire$(X_{near}, E, \mathbf{x}_{new})$
23: **end if**
24: **if** InGoalRegion$(\mathbf{x}_{new})$ **then**
25: $X_{soln} \leftarrow X_{soln} \cup \{\mathbf{x}_{new}\}$;
26: **end if**
27: **end for**
28: **return** (V, E)

(Gammell et al., 2014b) for more details and proofs about the ellipsoid that plays the major role in this algorithm.

Properties

What follows is a summary of properties of the RRT* algorithm. As mentioned earlier, informed RRT* inherits its asymptotic optimality.

Algorithm 6 Sample

Input: c_{max}, c_{min}, $\mathbf{x}_{centre}$, $\mathbf{C}$
if $c_{max} < \infty$ **then**
  $r_1 \leftarrow c_{max}/2$
  $\{r_i\}_{i=2,\dots,d} \leftarrow \left(\sqrt{c_{max}^2 - c_{min}^2}\right)/2$
  $\mathbf{L} \leftarrow$diag(r)
  $\mathbf{x}_{ball} \leftarrow$ SampleUnitBall
  $\mathbf{x}_{rand} \leftarrow (\mathbf{CLx}_{ball} + \mathbf{x}_{centre})$
  return $\mathbf{x}_{rand}$
end if
return $\mathbf{x}_{rand} \sim$ UniformSample(X)

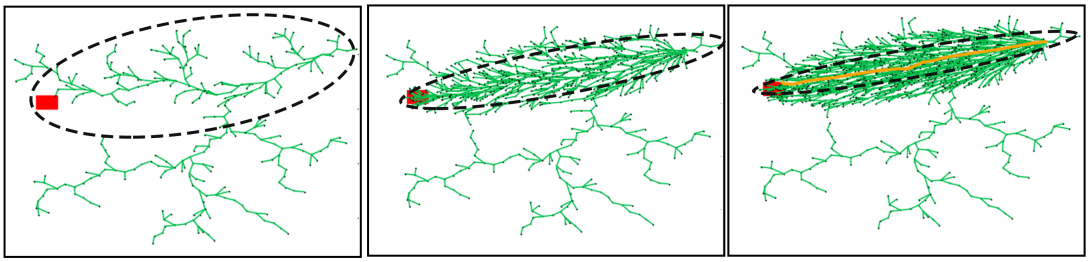

Figure 15. A run of Informed RRT* with $X_{free} = [0, 10] \times [0, 10]$ with a stepsize $\epsilon = 0.2$, $x_{init} = (8, 9)$ and $K = 1600$. The goal area is shown in red and the found path is shown in orange. The samples are drawn from within the ellipsoid and it shrinks as the solution converges to the optimal solution.

1. As the difficulty of the search space increases, Informed RRT* seems to improve the solution with an order-of-magnitude.
2. The algorithm converges to an optimal solution in finite time.
3. The algorithm improves on the solution without requiring extra parameters that need fine tuning.
4. Informed RRT* explores only regions capable of improving the solution and it does this with similar computation costs with respect to RRT*.

The first two properties are a theoretical improvement of RRT* by improving only the paths between the start and the goal. As opposed to RRT*-Smart that needs fine tuning of the biasing radius, Informed RRT* does not include more parameters to RRT*. Informed RRT* takes samples from the ellipsoid uniformly and does not violate the assumption of uniform density.

3.3.2. Problems

Informed RRT* improves on the convergence problem of RRT* but it has difficulty scaling well in higher dimensions. The heuristic sampling space becomes too large and the algorithm loses focus (L. Janson & Pavone, 2015). Since Informed RRT* uses RRT* to find a first solution, it is not practical for use in non-holonomic environments where a fast first solution is often needed (Daniel et al., 2014). Another disadvantage is the difficulty in complex planning problems, i.e., state spaces with difficult passages to the goal such as narrow corridors or tunnels.

Variants improving on Informed RRT*

In this subsection we provide a brief overview of a few algorithms that overcome some of the problems of Informed RRT*. Note that some algorithms could be different from Informed RRT*, while still improve on its problems. Some of these algorithms, such as the so called RABIT*, offer significant improvements in both simulated environments and in real world problems, however, an in-depth look at these is beyond the scope of this paper. The reader is invited to take a look at these. What follows is a list of algorithms improving on Informed RRT*:

- **BI^2 RRT*** is proposed in (Burget et al., 2016) for complex manoeuvring tasks such as moving a cart in a parking lot or the transport of liquids. The proposed algorithm tries to find a faster solution by employing two Informed RRT* trees. It therefore uses two ellipsoids from which to draw samples. This way, BI^2 RRT* achieves a faster convergence and it is shown that it does this in less planning time.
- **BIT***, or Batch Informed Trees (Gammell et al., 2014a), also tries to find fast a first solution in non-holonomic environments. It keeps the ellipsoid from the Informed RRT* algorithm but instead of gradually shrinking the ellipsoid as better paths are found, it starts with a small volume and enlarges the ellipsoid gradually. BIT* combines graph-search techniques with batches of samples to search in order of decreasing solution cost in much the same way A* searches a graph while being optimal. BIT* performed better than Informed RRT* in both simulated environments and in a problem domain of a two-armed robot.
- **RABIT*** (Regionally Accelerated BIT*) (Choudhury et al., 2016) is a hybrid algorithm combining optimisation techniques with BIT* and uses local information (cost gradients) to accelerate the search. It improves the problem of narrow passages of the Informed RRT* algorithm. It converged 1.8 times faster than BIT* in simulated environmens and in real world flight tests of an autonomous helicopter. It also improved significantly in higher dimensions (eight or more).

3.3.3. Conclusion

Informed RRT* goes through two stages. in the first stage the regular RRT* algorithm finds a goal. In the second stage an ellipsoid is created between the start node and the found goal node; this is the heuristic space. The samples are taken uniformly from the heuristic space

to focus the growth solely on the paths that can lead to an improvement. Informed RRT* has shown an order-of-magnitude improvement when no obstacles exist between the start and the goal. The algorithm converges to an optimal solution in finite time, whereas RRT* generally needs infinite time. The rotation matrix for the ellipsoid needs to be computed only once so that not a significant amount of computation is required to focus the search. However, Informed RRT* has difficulty focusing its search in higher dimensions, is often not fast enough to find a first solution and has trouble with narrow passages.

4. Experiments

In this section the behaviour of the three base algorithms are compared. RRT, RRT* and Informed RRT* were run on two simple planning problems. Their variants were not considered during these simulations. Moreover, This section serves as a demonstration of the properties of the three algorithms. It is not intended to be a fully fledged and complete comparison of the algorithms.

4.1. Experimental Setup

To demonstrate how the algorithms behave in an environment without obstacles, a 10×10 scenario was considered. The starting point for all the algorithms $\mathbf{x}_{init}$ was (5,5) and the goal area was a rectangle originated at (1,9). A step-size ϵ of 0.15 was chosen, given the dimensions of the plane. For RRT* and Informed RRT* the γ value was set to 50 and the maximum radius η was set to 0.4. The simulation of the first scenario was performed with the same pseudo-random seed for all the three algorithms. The convergence properties were also examined for this first scenario. The three algorithms were analysed using Monte-Carlo runs. The algorithms were run for 10000 iterations over 100 trials. For each iteration the cost of the best solution found was averaged.

The second scenario considered was also a 10×10 plane but with randomly generated obstacles. Some obstacles were chosen such that a narrow gap exists for the true shortest path between x_{init} and x_{goal}. The starting point for all algorithms $\mathbf{x}_{init}$ was (5.5,1) and the goal area was a rectangle originated at (6.8,8). The step-size ϵ, the constant γ and the maximum radius η were identical to the first scenario. The simulation of the second scenario was performed with the same pseudo-random seed for all the three algorithms. To demonstrate the behaviour of the algorithms and observe the convergence, the algorithms were first run for 20000 iterations. Additionally, in accordance with the first scenario, the convergence of the three algorithms was analysed using Monte-Carlo runs. The algorithms were run for 10000 iterations over 100 trials. For each iteration the cost of the best solution found was averaged.

4.2. Results

The convergence results are shown in Figure 17. The iterations are plotted against the cost of the best solution found. Figure 18 and 16 show the environments that were considered, the generated trees are coloured black, blue and green for RRT, RRT* and Informed RRT* respectively. Table 1 shows the results of the comparison between the three algorithms in

the second scenario.

4.2.1. First Scenario

In the first scenario it is shown that RRT does not try to improve its jagged path to the goal (Figure 18). Most of the time, RRT keeps the first found path, whereas RRT* and Informed RRT* do improve the path with the Parent and Rewire procedures to eventually converge to an optimal solution, which in this case is a straight line between the start and the goal. Informed RRT* uses the ellipsoid (the heuristic space) to converge much faster than RRT*. In Fig 17 it is seen that in the scenario without obstacles the cost of the best path to the goal for RRT stays constant. In the 100 runs RRT could not find the optimal path. RRT* and Informed RRT* approach the same optimal value but Informed RRT* does this on average before iteration 2000, in contrast, RRT* has not fully converged to the optimal path after 10000 iterations. In terms of best solution cost, in Figure 16 (b), (f) and (j) the compared efficiency is evident.

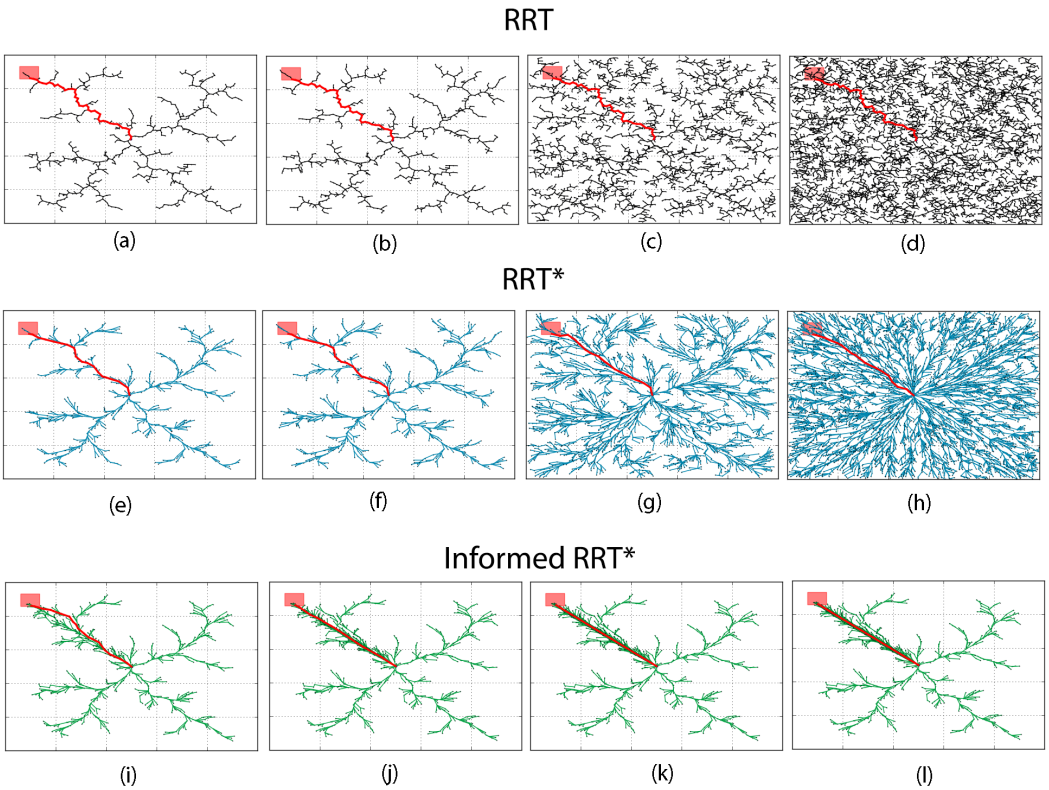

Figure 16. The three algorithms were analysed on a plane with no obstacles. The red path is the best found path. From left to right, the number of iterations at each frame is 600, 750, 2500 and 5000 respectively.

4.2.2. Second Scenario

The second scenario imposes a problem for the algorithms: the optimal path is through a narrow gap. Similar to the first scenario, in Figure 18 it is seen that RRT keeps the first found path, while RRT* and Informed RRT* gradually improve the best path. In the case of RRT*, the whole state space is considered. In this run RRT* finds a path through the narrow gap at iteration number 8272, whereas Informed RRT* found a better path through the narrow gap much earlier; at iteration number 3241. In the 100 Monte Carlo runs (Figure 17) RRT never finds a path to the goal through the narrow gap, while Informed RRT* always does and RRT* did 93% of the time within the 10000 iterations.

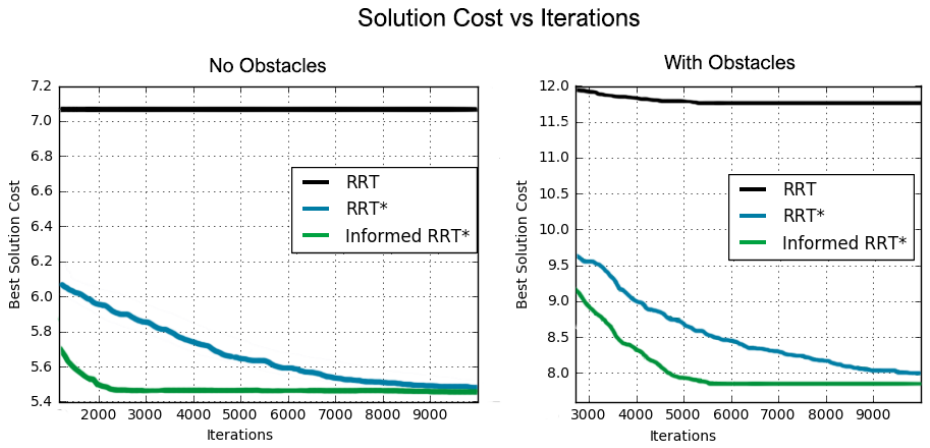

Figure 17. The solution cost versus iterations for RRT, RRT* and Informed RRT*. The three algorithms were run for 10000 iterations for 100 times.

Table 1. A comparative overview of the results in the second scenario.

	RRT	RRT*	Informed RRT*
Path Cost	10.69	8.09	7.85
Run Time (sec)	270	549	636
Expanded Nodes	8192	7435	6613
Optimality	Suboptimal	Optimal	Optimal

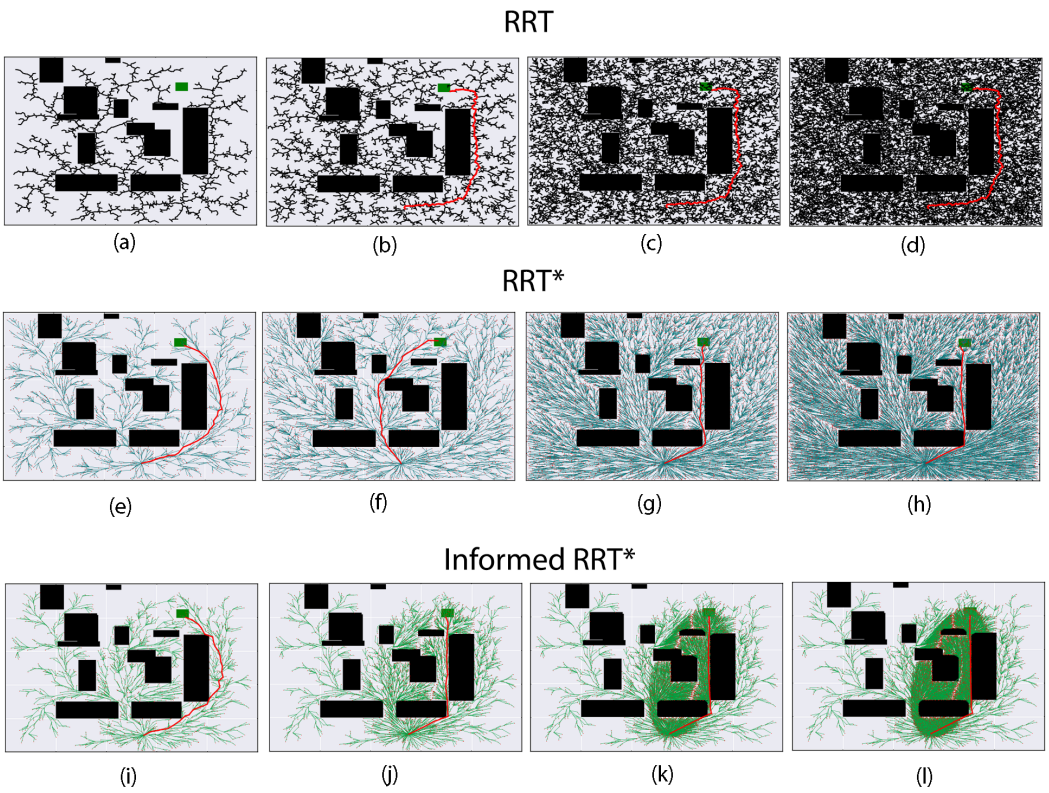

Figure 18. The three algorithms were analysed on a plane with obstacles. The red path is the best found path. From left to right, the number of iterations at each frame is 2500, 5000, 12500 and 20000 respectively.

5. Discussion

In this paper the theory behind the class of RRTs is discussed along with a summary of the recent developments regarding this sampling-based method. A basic understanding for three baseline algorithms (RRT, RRT* and Informed RRT*) is provided and is accompanied with a brief overview of the variants improving on these algorithms. Additionally, a comparison between RRT, RRT* and Informed RRT* is performed to demonstrate the capabilities and convergence properties of the three algorithms. It is shown that RRT produces jagged paths and always converged to a suboptimal path. It is also shown that both RRT* and Informed RRT* converge gradually to the optimal path but Informed RRT* does this significantly faster. Furthermore, RRT is incapable of finding paths through narrow passages. Both RRT* and Informed RRT* were able to find a path through the narrow passage but they still had trouble in doing so. Informed RRT* always managed to find the path through the narrow passage, while RRT* needed more time to converge to this path because it considers

the whole state space after it has found a path. Informed RRT* shows promising results in terms of convergence and behaviour. It proposes a method to handle the slow convergence of RRT*, while modifying it minimally. RRT* generally needs an infinite amount of time to converge to the optimal path, in contrast, Informed RRT* does this in finite time making the search for the optimal path a feasible one. But in this paper only the convergence to the path cost in terms of iterations was considered and only the two planning domains were used to examine this convergence. The three algorithms have the same time complexity, nevertheless, it is important to accentuate the differences in computational time. Therefore a comparison for computational time is suggested in different planning domains, since in some planning problems speed could be important.

It is discussed how other methods handle the slow convergence problem. RRT*-Smart smooths its path and uses an intelligent sampling heuristic but it is unknown how it performs against Informed RRT*. RABIT* was also briefly discussed and claims a significant improvement on Informed RRT* but this algorithm lies beyond the scope of this paper. To see how well these perform, a thorough comparison is suggested. In (Choudhury et al., 2016) a significant improvement over BIT* is claimed but RABIT* uses optimisation techniques along with BIT*. In some simple planning domains it could be unnecessary to perform optimisation techniques along with matrix operations. Informed RRT* could in some cases be sufficient. When only a small number of nodes is allowed, RRT could suffice to find a first path rapidly in terms of computation time. Moreover, we suggest a simple improvement to the Informed RRT* algorithm by adapting the path optimisation procedure from RRT*-Smart to the algorithm. It is thought that this will essentially smoothen the path to the goal by the triangular inequality. The nodes in the optimised path need not function as beacons for intelligent sampling because it is in a sense replaced by the ellipsoid from Informed RRT*.

A key advantage of the RRT class over regular (exact) algorithms is that it could consider (non-)holonomic constraints with each node expansion. Its incremental and probabilistic nature allows this to happen faster than exact algorithms. In this paper non-holonomy was only discussed briefly but this property is the main reason RRT was invented. For future work a detailed account of non-holonomy in the context of RRTs is suggested, since it usually requires an application of differential constraints or concepts from advanced linear algebra. Furthermore, in this paper a static two-dimensional environment is considered for explanation and experimentation. The real world is dynamic and is inherently uncertain, these are two concepts that need more work in the context of RRTs. Although these are discussed to some extent in (LaValle, 2006), the author strives for a general explanation in the context of planning algorithms and not specifically in an RRT environement. Furthermore, (LaValle, 2006) lacks an updated account on the development of RRTs. In this paper we discuss RRT, RRT* and Informed RRT* and some of their variants but the class of RRT is rapidly growing and many variants exist, taking into account a wide variety of motion planning problems. Nonetheless, this paper lays down the basic concepts behind RRTs and provides a brief overview of the class.

5.1. Conclusion

In this paper the concepts of RRTs is explained and three baseline algorithms are discussed: RRT, RRT* and Informed RRT* along with their variants. We discuss their behaviour

and some of the theory behind the algorithms. A comparison between these algorithms is performed to examine this behaviour and their rate of convergence. Further experimental work is necessary to investigate their speed performance. Our results suggest that Informed RRT* converges much faster than RRT* and that RRT keeps a chaotic and jagged first solution. It should be noted that no updated account on the class of RRTs exists and that, considering the many variations and problems, a thorough research is needed.

6. Acknowledgments

First and Foremost I would like to thank my advisor Gerard Vreeswijk, whose suggestions, tips and tricks on academic writing helped me more than he can imagine. I would like to thank him as well for the subject. It opened a door for me to new and fascinating developments in motion planning. I would also like to thank those who reviewed bits and pieces of this paper: Vivian and Anna for their extensive review sessions, Sander Bos for his imaginative title suggestions and Rutger Teeuwen for his almighty Choobness.

7. Appendix

7.1. Extra Information

As mentioned before, RRTs are not only for use in Motion Planning they have been extensively been used for modelling Diffusion Limited Aggregation (Witten & Sander, 1981). See Figure 19 (a) for an image of DLA. In DLA particles cluster together by performing a random walk and form tree-like structures. DLA has been observed in electrical breakdowns and mineral deposits. Compare the image with Figure 5. A note worthy paper is (Burch & Weiskopf, 2013) where the authors take an aesthetic approach and show that RRT can also be used for art purposes (Figure 19).

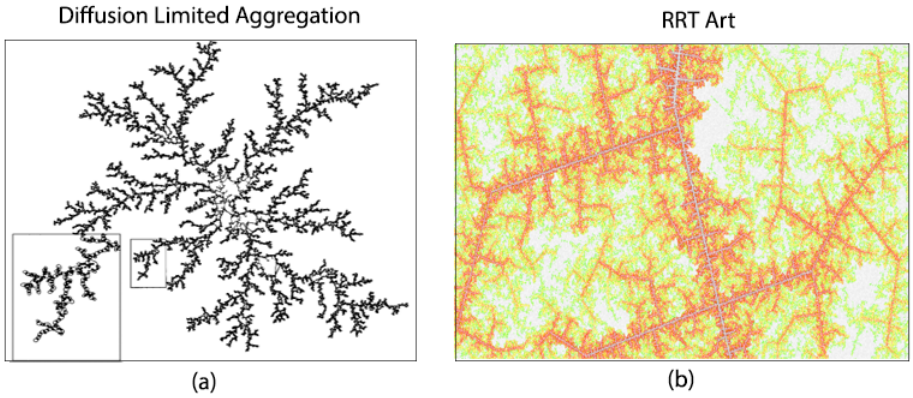

Figure 19. Frame (a) shows a generated cluster (DLA) (Witten & Sander, 1981) and frame (b) shows one million iterations of the regular RRT algorithm (Burch & Weiskopf, 2013)

7.2. The Implementation

The algorithms were implemented using Python for its ease of use and extensive libraries that support matrix operations and scientific computing. The implementation of the three algorithms is found on https://github.com/Cynetics/RRTs.

7.3. The Images

All the images were made by Bryan Cardenas Guevara and generated with Photoshop CSS 6 to illustrate the examples and provide an ease of understanding. These images are free to use and are also found on https://github.com/Cynetics/RRTs.

References

Adiyatov, O. and Varol, H. A. Rapidly-exploring random tree based memory efficient motion planning. *presented at the IEEE International Conference of Mechatronics and Automation (ICMA)*, 2013.

Akgun, Baris and Stilman, Mike. Sampling heuristics for optimal motion planning in high dimensions. *In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*, 2011.

Arslan and Tsiotras, P. "use of relaxation methods in sampling-based algorithms for optimal path planning". *In IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany.*, 2013.

Arya, S., Mount, D. M., Silverman, R., and Wu., A. Y. An optimal algorithm for approximate nearest neighbor search in fixed dimensions. *Journal of the ACM, 45(6):891–923 .*, 1999.

Bentley, Jon. Multidimensional binary search trees used for associative searching. *Association for Computing Machinery, Stanford*, 1975.

Bertsekas, D. P. Convergence of discretization procedures in dynamic programming. *TAC, 20(3): 415–419*, 1975.

Burch, Michael and Weiskopf, Daniel. The aesthetics of rapidly-exploring random trees. *VISUS, University of Stuttgart*, 2013.

Burget, F., Bennewitz, M., and Burgard, W. Bi2rrt*: An efficient sampling-based path planning framework for task-constrained mobile manipulation. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

Chazelle, B. Triangulating a simple polygon in linear time. *Discrete and Computational Geometry 6(5):485–524*, 1991.

Choudhury, Sanjiban, Gammell, Jonathan D., Barfoot, Timothy D., Srinivasa, Siddhartha S., and Scherer, Sebastian. Regionally accelerated batch informed trees (rabit*): A framework to integrate local information into optimal path planning. *IEEE International Conference on Robotics and Automation (ICRA) Stockholm, Sweden, May 16-21*, 2016.

D. Kim, J. Lee and e. Yoon, S. Cloud rrt: Sampling cloud based rrt. Technical report, IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, 2014.

Daniel, Kenny, Nash, Alex, Koenig, Sven, and Felner, Ariel. Theta*: Any-angle path planning on grids. 2014.

Donald, Bruce, Xavier, Patrick, Canny, John, and Reif, John. Kinodynamic motion planning. *Journal of the Association for Computing Machinery. Vol 4, No 5, pp 1048-1066*, 1993.

Fu, Cong and Cai, Deng. EFANNA : An extremely fast approximate nearest neighbor search algorithm based on knn graph. *CoRR*, 2016.

G., Peano. Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen (in French), 36 (1): 157–160*, 1890.

Gammell, Jonathan D., Srinivasa, Siddhartha S., and Barfoot, Timothy D. Bit*: Batch informed trees for optimal sampling-based planning via dynamic programming on implicit random geometric graphs. *CoRR,abs/1405.5848*, 2014a.

Gammell, Jonathan D., Srinivasa, Siddhartha S., and Barfooti, Timothy D. Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. *In IEEE/RSJ International Conference on Intelligent Robots and Systems.*, 2014b.

Hossam, El Gindy and David, Avis. "a linear algorithm for computing the visibility polygon from a point". *Journal of Algorithms. 2 (2): 186–197. doi:10.1016/0196-6774(81)90019-5.*, 1981.

J.H.Reif. Complexity of the piano mover's problem and generilizations. *In Proceedings IEEE Symposium on Foundations of Computer Science, pages 421–427*, 1979.

Johnson, B.D. The nonholonomy of the rolling sphere. *American Mathematical Monthly. vol 114, 500*, 2007.

Jordan, Matthew and Perez, Alejandro. Optimal bidirectional rapidly-exploring random trees. *Computer Science and Artificial Intelligence Laboratory Technical Report .*, 2013.

Jouandeau, N. and Cherif, A. Ali. Fast approximation to gaussian obstacle sampling for randomized motion planning. Technical report, 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles Instituto Superior Técnico, Lisboa, Portugal, 2004.

Karaman, S. and Frazzoli, E. Sampling-based motion planning with deterministic μ-calculus specifications. *In IEEE Conference on Decision and Control (CDC)*, 2009.

Karaman, S. and Frazzoli, E. Sampling-based optimal motion planning for non-holonomic dynamical systems. *2013 IEEE International Conference on Robotics and Automation, 5041-5047*, 2013.

Karaman, S. and Frazzoli, Emilio. Incremental sampling-based algorithms for optimal motion planning. Technical report, Robotics Science and Systems VI 104, 2010.

Karaman, S., Walter, M., Perez, A., Frazzoli, E., and Teller, S. Anytime motion planning using the rrt*. *presented at the IEEE International Conference on Robotics and Automation (ICRA).*, 2011a.

Karaman, Sertac and Frazzoli, Emilio. Sampling-based algorithms for optimal motion planning. *CoRR*, abs/1105.1186, 2011.

Karaman, Sertac, Walter, Matthew R., Perez, Alejandro, Frazzoli, Emilio, and Teller, Seth. Anytime motion planning using the rrt*. *In IEEE International Conference on Robotics and Automation (ICRA) May 9-13, Shanghai International Conference Center, Shanghai, China.*, 2011b.

Kavraki, L. E., Svestka, P., Latombe, J.-C., and Overmars, M. H. "probabilistic roadmaps for path planning in high-dimensional configuration spaces". *IEEE Transactions on Robotics and Automation, 12 (4): 566–580*, 1996.

Kim, Min-Cheol and Song, Jae-Bok. Informed rrt* with improved converging rate by adopting wrapping procedure. *Intelligent Service Robotics*, 2018.

Kuffner, James J. and LaValle, S. M. Rrt-connect: An efficient approach to single-query path planning. Technical report, Computer Science Dept. Stanford University, CA 94305 USA, 2000.

L., A. Sanchez and Arenas, J. Abraham. Nonholonomic motion planning for car-like robots. *MICAI 2002: Advances in Artificial Intelligence pp 1-10*, 2002.

L. Janson, E. Schmerling, A. Clark and Pavone, M. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *IJRR, 34(7): 883–921*, 2015.

Lan, X. and Cairano, S. Di. Continuous curvature path planning for autonomous vehicle maneuvers using rrt*. *presented at the European Control Conference (ECC).*, 2015.

Latombe, J.C. "robot motion planning". *Kluwer Academic Publishers Boston, United States.*, 1991.

LaValle, S.M. Rapidly-exploring random trees: A new tool for path planning. Technical report, Computer Science Department, Iowa State University, 1998.

LaValle, Steven M. (ed.). *Planning Algorithms*. Cambridge University Press, University of Illinois, 2006.

LaValle, Steven M. and Kuffner, James J. Rapidly-exploring random tree: Progress and prospects. *Algorithmic and Computational Robotics: New Directions,*, 2000.

Lee, H. Song and Shim, D. H. Optimal path planning based on splinerrt* for fixed-wing uavs operating in three-dimensional environments. *presented at the 14th International Conference on Control, Automation and Systems (ICCAS), Korea*, 2014.

Li, Tsai-Yen and Shie, Yang-Chuan. An incremental learning approach to motion planning with roadmap management. Technical report, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Taipei Taiwan, 2002.

Lindemann, Stephen R. and LaValle, S. M. Incrementally reducing dispersion by increasing voronoi bias in rrts. 2004.

Moon and Chung, W. Kinodynamic planner dual-tree rrt (dt-rrt) for two-wheeled mobile robots using the rapidly exploring random tree. *IEEE IND ELECTRON, vol. 62*, 2015.

Nasir, Jauwairia, Islam, Fahad, Malik, Usman, Ayaz, Yasar, Hasan, Osman, Khan, Mushtaq, and Muhammad, Mannan Saeed. Rrt*-smart: A rapid convergence implementation of rrt*. *International Journal of Advanced Robotic Systems*, 10(7):299, 2013. doi: 10.5772/56718.

Overmars, P. Svestka M. H. Probabilistic path planning. *Laboratoire d'Analye et d'Architecture des Systemes Centre National de la Recherche Scientifique LAAS report 97438*, 1998.

P. E. Hart, N. J. Nilsson and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *TSSC, 4(2): 100–107*, 1968.

Professor Emilio Frazzoli, Sertac Karaman, hwan Jeon, Jeong, Teller, S., Walter, M., and Kim, Been. Robust planning and control for agile robotics for logistics. *(ICRA), for US Army Logistics Innovation Agency (LIA) .*, 2009.

Schwartz, J.T. and Sharir, M. On the "piano-movers" problem: I. the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics, 36(3):345–398*, 1983.

Sujith Kumar, B, S Abhimanyu, P, V V P Bharagav, B, Agarwal, Pratik, and Krishna, Madhava. Robocup ssl team description, irl rc. Technical report, International Institute of Information Technology, Hyderabad, 2010.

Swaczyna, Martin. Several examples of nonholonomic mechanical systems. *Communications in Mathematics 19 27–56*, 2011.

Urmson, JChris and Simmons, Reid. Approaches for heuristically biasing rrt growth. *Proceedings of the 2003 IEEElRSJ InU. Conference on Intelligent Robots and Systems Las Vegas, Nevada. October 2003*, 2003.

Witten, T. A. and Sander, L. M. Diffusion-limited aggregation, a kinetic critical phenomenon. *Physical Review Letters. 47 (19): 1400–1403.*, 1981.