AN ANALYSIS OF PATH PLANNING ALGORITHMS

University of Maryland, College Park

CMSC351H Final Project

Bo He

December 11, 2022

Contents

Abstract

This paper is a basic analysis of path planning algorithms. This is the final project for class CMSC351H taken at the University of Maryland, College Park at Fall 2022.

This paper first gives a brief analysis of Artificial Intelligence and its subsequent development by an undergraduate student, introducing what AI is, autonomous driving, and the connection between them. Then, it will present a study of different path planning algorithms and their applications in various fields. The author provide an overview of the different types of path planning algorithms, including $A*$ search, Dijkstra's algorithm, and the RRT algorithm. The paper concludes with a discussion of the strengths and limitations of each algorithm, as well as potential future research directions. The paper is for class use and self-study only.

1 Introduction to Artificial Intelligence

> "The development of full artificial intelligence could spell the end of the human race. . . .It would take off on its own, and re-design itself at an ever increasing rate. Humans, who are limited by slow biological evolution, couldn't compete, and would be superseded."
>
> — Stephen Hawking told the BBC

The development of technology has really brought qualitative changes in the quality of life for human beings. Some of the most notable changes include increased connectivity and communication, enhanced transportation and infrastructure, and the development of new and improved medical treatments and technologies. Talking more to the technology of electronics, It is possible and easy for people to access a vast amount of information and has greatly facilitated the ability to share ideas and knowledge through the internet. It has also made many everyday tasks easier and more convenient. Besides, with the popularity of computers and the explosion of technology, artificial intelligence(AI) is often talked about. Artificial intelligence refers to the simulation of human intelligence in machines that are programmed to think and act like humans. These machines are designed to learn from their experiences, adjust to new inputs and perform human-like tasks. The changes brought by the development of AI to society are more easily detected, including increased automation and efficiency in various industries, the ability to analyze and process large amounts of data quickly and accurately, and the development of new and improved technologies and products.[1] AI has also led to the creation of new job opportunities in fields such as machine learning and data science. Additionally, it has the potential to improve healthcare, transportation, and other areas of our lives. However, it has also raised concerns about job displacement and the ethical implications of developing and using AI. As Stephen Hawking put it, "The development of full artificial intelligence could spell the end of the human race.". There are fears that the development of AI will bring with it a sense of autonomy, with incalculable consequences. However, based on my understanding, for the time being, AI is far more helpful to human society than it is potentially harmful. Overall, the development of AI has brought both opportunities and challenges.

AI has its roots in several older disciplines: Philosophy, Logic, Computation, Cognitive Science/Psychology, Biology/Neuroscience, and Evolution. By looking at each of these in turn, we can gain a better understanding of their role in AI, and how these underlying disciplines have developed to play that role. Major sub-fields of AI now include: Machine Learning, Neural Networks, Evolutionary Computation, Vision, Robotics, Expert Systems, Speech Processing, Natural Language Processing, and Planning. The paper will give a detailed description for some areas in the following sections.

1.1 Machine Learning

Machine learning is a subset of artificial intelligence that involves the use of algorithms and statistical models to enable a system to improve its performance on a specific task through experience. It has become an increasingly important area of research and development, with applications ranging from image and speech recognition to natural language processing and predictive analytics. The field of machine learning has seen significant growth in recent years, with a wide range of algorithms and techniques being developed to tackle a variety of tasks. Some of the most commonly used algorithms include decision trees, support vector machines, and neural networks. These algorithms have been applied to a wide range of domains, including computer vision, natural language processing, and predictive modeling. One of the key challenges in the field of machine learning is the need

for large amounts of high-quality training data. This can be difficult to obtain, particularly for specialized or niche applications. Additionally, there are concerns about the ethical implications of using machine learning, particularly in areas such as decision-making and automated systems.[3]

Despite these challenges, the potential benefits of machine learning are considerable. It has the potential to improve the accuracy and efficiency of many tasks, and to enable the development of new applications and technologies. In the coming years, it is likely that machine learning will continue to play a key role in the development of artificial intelligence and other advanced technologies. Overall, the field of machine learning is an exciting and rapidly-evolving area of research. It has the potential to impact a wide range of domains, and its developments will be closely watched by researchers and practitioners alike.

1.2 Robotics

AI Robotics is an emerging field of research which aims to develop artificial intelligence (AI) techniques and algorithms that can be used in robotics systems. AI Robotics focuses on the design, development, and use of intelligent robots that can perform useful tasks without direct human supervision. The main goal for AI Robotics researchers is to create a system that has the same or similar capabilities as humans with respect to perception, reasoning, learning ability and interaction with their environment. The main purpose of developing such systems is not only to make these robots more efficient but also help people understand how they work so they can improve their own performance. For example: by creating a robot capable of making decisions based on its understanding of the world it will help us better understand our own cognitive abilities.[4]

What are some examples from everyday life? There are many real-world applications where AI robotics technologies have already been implemented successfully including prosthetic limbs for amputees; automated checkout counters in retail stores; robotic arms for surgery; robotic aircrafts like drones; autonomous vehicles like cars or trucks etc.; service robots for elderly care facilities etc.; home automation devices like smart speakers at home; self-driving cars etc.; also there are many other exciting areas where new developments are happening rapidly such as: virtual assistants like Siri or Alexa from Apple or Google Assistant from Alphabet Inc.. These technologies may soon become part of our daily lives!

1.3 Evolutionary Computation

Evolutionary Computation (EC) is a method for solving optimization problems. It has been used in many areas, such as robotics and bioinformatics. The basic idea of EC is to simulate the evolution of populations of agents: these are simple programs that can be run on computers or simulated by an algorithm. In order to evolve them, we need some starting conditions for each agent and the rules for their interaction with other agents. [5] In practice, this means that we need to specify the fitness function (the measure of how well an agent performs its task), so that it evolves towards being better at performing its tasks than any other agent in the population. We also need rules describing how two different agents interact: when one learns from another, which gives it more knowledge about what it needs to do next time; when two agents fight against each other; etc. Evolutionary computation is not limited to simulating biological systems, but can be applied to a wide range of problems from designing artificial life-forms (e.g., robots) or even creating virtual worlds with intelligent agents living within them ! In fact, it has been argued that evolutionary computation is fundamentally different from classical mathematical methods because this new ap-

proach does not longer rely on the notion of a "first principle" to guide its search for solutions, but rather it uses an iterative process that is guided by the data gathered and tested during previous runs.[5]

So far so good! But there are several key concepts behind EC which make things more complex than they might seem at first glance: First, because our goal is to evolve systems rather than just optimize functions or solve equations exactly, we have no way of knowing beforehand whether our solution will be useful in real life – i.e., whether it will be able to perform effectively any given task under actual conditions – let alone all possible ones![6] Second, unlike most classic computational models based on specific types of physical processes in nature (such as Newton's laws or Maxwell's equations), EC does not assume anything about the underlying physics – i.e., no matter what kinds of particles/agents/etc., we use as building blocks for our simulation model, they will always obey certain fixed laws governing their interactions with each other and with their environment. This means that although classical mathematical techniques have been developed over centuries based on assumptions about physical reality, they do not apply directly here since there are no known natural laws underlying biological phenomena! Instead, scientists must usually develop new theories appropriate specifically for biological systems. [6]

1.4 Natural Language Processing

Natural language processing (NLP) is a field of artificial intelligence that focuses on enabling computers to understand, interpret, and generate human language. This technology has a wide range of applications, including language translation, chatbots, and text analysis. At a high level, NLP involves three main tasks: understanding the structure of a language, determining the meaning of words and sentences, and generating natural-sounding language. To accomplish these tasks, NLP systems use a combination of rule-based and statistical methods. Rule-based NLP systems rely on a set of predefined rules and algorithms to analyze language. These systems can be effective for tasks like grammar checking and spelling correction, but they can be limited in their ability to understand the meaning of language and handle complex or ambiguous sentences. Statistical NLP systems, on the other hand, use machine learning algorithms to analyze large amounts of language data and make predictions about the meaning of words and sentences. These systems can be more flexible and adaptable than rule-based systems, but they require large amounts of training data to perform well.[8]

NLP has seen significant progress in recent years. One of the key drivers of this progress has been the increasing availability of large amounts of data, which has allowed NLP algorithms to be trained on more diverse and nuanced language examples. Another important factor has been the development of more advanced machine learning techniques, such as deep learning, which has improved the performance of NLP systems on a wide range of tasks. This has led to NLP systems that are more accurate, versatile, and able to handle complex or ambiguous language inputs. However, as for now, the challenge for this technology is the need for NLP systems to be able to handle a wide range of languages and dialects. While some NLP systems are designed to work with a specific language, there is a growing demand for systems that can handle multiple languages, which requires more sophisticated algorithms and more data for training.

Overall, the field of NLP has made great strides in recent years, but there is still much work to be done to fully understand and replicate the complexity of human language.

1.5 Computer Vision

Computer vision is a field of artificial intelligence that focuses on enabling computers to interpret and understand visual data from the world around them. This technology is used in a wide range of applications, including image and video recognition, object detection, and autonomous navigation. Talking about the operation mode, computer vision systems use a combination of sensors, algorithms, and machine learning to analyze visual data. The input to a computer vision system is typically one or more images or video streams, which are processed to extract relevant information and make predictions about the objects or scenes depicted in the images. There are many different approaches to computer vision, but most systems can be broadly classified into two categories: rule-based and machine learning-based. Rule-based systems use a set of predefined rules and algorithms to interpret visual data, while machine learning-based systems use training data to build predictive models that can recognize patterns in the data.[9]

Computer vision is a rapidly advancing field of artificial intelligence that has seen significant progress in recent years. This progress has been driven by several factors, including the availability of large amounts of data, the development of more powerful hardware, and the advancement of machine learning algorithms. One of the key challenges in computer vision is the need to handle a wide range of lighting conditions, angles, and other factors that can affect the appearance of objects in images. Another challenge is the need to process large amounts of data in real-time, which requires efficient algorithms and powerful hardware. To address these challenges, we need use a combination of different algorithms and techniques to allow a quicker response. Another way is developing more powerful hardware to allow system make decisions quickly and accurately.

Overall, the field of computer vision has made great strides in recent years, but there are still many challenges to be addressed. These challenges include improving the accuracy and robustness of computer vision systems, as well as developing more efficient algorithms and hardware.

2 Autonomous Driving

Autonomous driving, also known as self-driving or driverless, refers to the ability of a vehicle to operate without a human driver. Autonomous vehicles use a variety of sensors and advanced technologies, such as artificial intelligence, to navigate and make decisions on the road. These vehicles are designed to be safer, more efficient, and more convenient than traditional vehicles. Path planning, as one of the main functions of autonomous driving, plays a very important role, or we can say that it is the cornerstone of autonomous driving and are essential for the development of self-driving vehicles. Autonomous driving to enable a vehicle to navigate and drive itself without the need for human input. This is typically done using a combination of sensors, maps, and algorithms that help the vehicle make decisions about how to navigate its environment. Path planning, on the other hand, is the process of determining the best route for the vehicle to follow to reach its destination.[10]

One of the key challenges in autonomous driving and path planning is the need to make decisions in real-time based on a constantly changing environment. This requires the use of advanced algorithms and sensors that can quickly and accurately perceive the vehicle's surroundings and determine the best course of action. Another challenge is the need to account for a wide range of factors that can affect the safety and efficiency of the vehicle's journey, such as the location of obstacles, traffic laws, and the vehicle's capabilities. This requires the use of sophisticated planning algorithms that can take into account these factors and generate a safe and efficient route.

Overall, the development of autonomous driving and path planning technology has the potential to greatly improve road safety and efficiency, as well as provide new mobility options for individuals who are unable to drive. However, there are still many challenges to be addressed, such as ensuring the reliability and robustness of these systems in a wide range of environments and conditions.

2.1 The 6 Levels of Vehicle Autonomy

The 6 levels of vehicle autonomy, as defined by the Society of Automotive Engineers (SAE)[7], are as follows:

- Level 0: No Automation - The driver is in complete control of the vehicle and all of its functions at all times.
- Level 1: Driver Assistance - The vehicle has one or more automated functions, such as electronic stability control or adaptive cruise control, but the driver must remain engaged and in control of the vehicle at all times.
- Level 2: Partial Automation - The vehicle has multiple automated functions, such as lane keeping assist and automatic emergency braking, but the driver must still monitor the driving environment and be prepared to take control of the vehicle if necessary.
- Level 3: Conditional Automation - The vehicle can operate autonomously in certain driving situations, such as on a highway, but the driver must be ready to take control of the vehicle if the system encounters a situation it cannot handle.
- Level 4: High Automation - The vehicle can operate autonomously in most driving situations, but the driver must still be able to take control of the vehicle if necessary.

- Level 5: Full Automation - The vehicle can operate autonomously in all driving situations and does not require a human driver to be present.

Each level of vehicle autonomy represents an increase in the capabilities of the vehicle's automated systems, with Level 0 being the least autonomous and Level 5 being fully autonomous. The levels are not meant to be a strict classification, and there may be overlap between adjacent levels.

2.2 Three Primary Sensors

In most autonomous vehicles, the combination of cameras, LiDAR, and RADAR form the primary set of sensors that provide imaging, detection, ranging, tracking, and sensing of the drive location for a seamless ride.[12]Autonomous vehicles ensure that no human intervention is required when driving with the use of sensors. Autonomous vehicles use a variety of sensors to develop trustworthy vision. The sensors enable the self-driving vehicle to operate without causing fatalities by allowing it to recognize obstacles or obstructions in the road environment.

In autonomous vehicles, various sensors work together to provide a precise detection system. The core set of sensors in the majority of autonomous cars combines cameras, LiDAR (light detection and ranging), and RADAR (radio detection and ranging) to give the functionality of imaging, detection, ranging, tracking, and sensing of the drive location for a smooth ride. These overlapping sensor functions aid in the detection of the three-dimensional shape, proximity, and speed of adjacent objects.

Let's discuss the three primary sensors that help autonomous vehicles function.

- **Camera Vision**

  The car can record photographs and videos of its surroundings using camera sensors. This can be used to find things, road signs, and other environmental elements. The best sensor for providing a precise visual picture of an autonomous vehicle's surroundings is a camera. To provide a 360° vision, cameras are installed on the front, back, right, and left sides of autonomous cars. These cameras sense both short-range wide vision and long-range arrow view using wide and narrow fields of view. In autonomous vehicles, super-wide lenses are employed to capture a panoramic image that helps in parking.[12] However, precise camera images are unable to provide information about the separation between objects and autonomous cars. Object distance from the vehicle can be calculated using LiDAR technology. Notably, Tesla is the only business on the market that makes use of a pure vision autopilot system.

- **LiDAR**

  These sensors use lasers to measure the distance to objects around the vehicle, creating a 3D map of the vehicle's environment. This allows the vehicle to accurately detect the location and movement of nearby objects. LiDAR measures the separation between two objects using laser beams (light waves). LiDAR is positioned on top of vehicles in autonomous vehicles and rotates quickly while shooting out laser beams. The laser beams return to the apparatus after being reflected by the obstructions. The autonomous vehicle's surroundings are measured in terms of distance, shape, and depth using the time it takes for this to occur. LiDAR can detect an obstacle's position, form, size, and depth, but it can also produce false echoes that misidentify nearby objects as far away ones and vice versa. LiDAR can't tell

between numerous laser signals at once and depicts barriers that don't exist for autonomous vehicles. Rain, snow, and fog all impair LiDAR performance. As a result, the best and most accurate sensor for autonomous vehicles is RADAR technology.[12]

- **RADAR**

  These sensors use radio waves to detect objects around the vehicle. This can be used to measure the distance, speed, and direction of objects, allowing the vehicle to accurately track their movements. LiDAR and RADAR operate on the same principles, although RADAR uses radio waves as opposed to LIDAR's use of light waves. The distance, angle, and velocity of an object in front of the autonomous vehicle are determined using the time it takes for radio signals to return from it to the device. Millimeter waves are used in autonomous car RADAR, which offers millimeter-level accuracy. Autonomous vehicle RADAR uses millimeter waves to provide centimeter-accurate location and movement determination in addition to high resolution obstacle recognition. In contrast to other sensor technologies, RADAR consistently operates in low visibility conditions, such as overcast weather, snow, rain, and fog, in autonomous cars.[12]

2.3 Path Planning

Path planning is a complex task that involves a number of different factors and considerations. One important aspect is the ability to create an accurate map of the vehicle's surroundings. This typically involves using sensors such as cameras and lidar to gather data about the environment, which is then processed by algorithms to create a detailed map. Once the vehicle has a map of its surroundings, it must use this information to plan a safe and efficient route to its destination. This involves considering factors such as the speed and direction of other objects, traffic rules, and the vehicle's own capabilities. The vehicle must also be able to update its path plan in real time as the environment around it changes. Another important aspect of path planning is the ability to handle uncertainty. Since the vehicle's surroundings are constantly changing, the path plan must be able to adapt to new information and unexpected events. This may involve using techniques such as probabilistic modeling to account for potential obstacles or uncertainties in the environment. Overall, the success of an autonomous vehicle's path planning system is determined by its ability to accurately map the environment, plan a safe and efficient route, and adapt to changes and uncertainties in real time.

3 Path Planning Algorithms

Path planning algorithms are used to find the shortest or most efficient path between two points in a given environment. These algorithms are commonly used in robotics, autonomous driving, and other applications where it is important to find the best way to navigate through a complex or changing environment. There are many different algorithms that can be used for path planning, but three of the most common ones are A* algorithm, Dijkstra's algorithm, and the Rapidly-exploring Random Tree(RRT) algorithm. In the following sections, I will give a briefly overview of each of them and their applications in various fields.

3.1 A* Search Algorithm

A* is well known for its use in video games, and with the development of artificial intelligence and technology, the A* algorithm has become one of the most common algorithms used in modern technology industry — autonomous driving. The A* search algorithm is an extension of the Dijkstra's algorithm(which we will be discussed next) that uses an additional heuristic function to guide the search and expedite the process of finding the shortest path between two points. This heuristic function estimates the cost of the cheapest path from the current node to the goal node. By using this estimated cost, the A* algorithm is able to intelligently search the game environment and quickly find the optimal path for the game character to follow.

3.1.1 Explanation

The A* algorithm is a type of search algorithm that is commonly used in computer science for finding the shortest path between two points. It combines the strengths of both the breadth-first search and the uniform-cost search algorithms, while avoiding their weaknesses.

The A* algorithm works by maintaining a list of nodes to be explored, called the open list, and a list of nodes that have already been explored, called the closed list. At each step, the algorithm chooses the node on the open list that has the lowest estimated total cost (i.e. the sum of the cost to reach that node and the estimated cost to reach the goal from that node) and expands it, adding any resulting child nodes to the open list if they are not already on the open list or on the closed list.

The key difference between the A* algorithm and other search algorithms is that it uses a heuristic function to estimate the cost to reach the goal from a given node. This heuristic function must be both admissible (i.e. it must never overestimate the cost to reach the goal) and consistent (i.e. the estimated cost to reach the goal from a given node must be less than or equal to the sum of the estimated cost to reach its successor plus the cost of the edge connecting the two nodes). Using this heuristic function, the A* algorithm can quickly discard many potential paths that would be considered by other search algorithms, allowing it to focus on the most promising paths and find the shortest path more quickly. This makes it a powerful tool for solving complex search problems.

3.1.2 Example

The examples are from geeksforgeeks, it can be found at here.

3.1.3 Pseudo Code[14]

```
function reconstruct_path(cameFrom, current)
    total_path := {current}
    while current in cameFrom.Keys:
        current := cameFrom[current]
        total_path.prepend(current)
    return total_path

function A_Star(start, goal, h)

    openSet := {start}

    cameFrom := an empty map
    gScore := map with default value of Infinity
    gScore[start] := 0

    fScore := map with default value of Infinity
    fScore[start] := h(start)

    while openSet is not empty
        current := the node in openSet having the lowest fScore[] value
        if current = goal
            return reconstruct_path(cameFrom, current)

        openSet.Remove(current)
        for each neighbor of current

            tentative_gScore := gScore[current] + d(current, neighbor)
            if tentative_gScore < gScore[neighbor]
                cameFrom[neighbor] := current
                gScore[neighbor] := tentative_gScore
                fScore[neighbor] := tentative_gScore + h(neighbor)
                if neighbor not in openSet
                    openSet.add(neighbor)

    return failure
```

3.2 Dijkstra's Algorithm

Dijkstra's algorithm is a popular algorithm used for finding the shortest path between two points in a graph. It is often used in network routing and other applications that require finding the optimal path between two points. The algorithm works by starting at the source node and exploring the neighboring nodes, updating their distances from the source node as it goes. The algorithm then proceeds to the next node with the smallest distance from the source and repeats the process until it reaches the destination node.

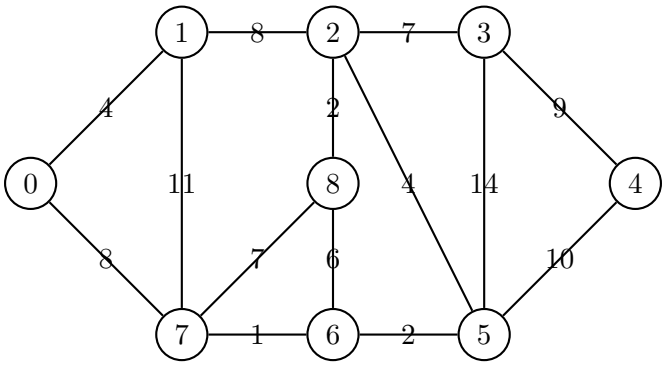
3.2.1 Explanation

In order to create a shortest path tree, Dijkstra's algorithm solves the single-source shortest path issue for a graph with non-negative edge weights. This algorithm is frequently used in routing and in other graph algorithms as a subroutine. Edsger W. Dijkstra, a computer scientist, came up with the idea in 1956, and it was published three years later.

The algorithm works by maintaining a set of nodes for which the shortest path from the source node is known. Initially, this set only contains the source node. The algorithm then iterates over the nodes in the graph, considering each unvisited node and relaxing the edges that connect it to other nodes in the graph. This process continues until all nodes have been visited and the shortest path to each one is known. One of the key features of Dijkstra's algorithm is its use of a priority queue to order the nodes that are candidates for exploration. This allows the algorithm to efficiently explore the nodes in the order that is most likely to produce the shortest path. Overall, Dijkstra's algorithm is a powerful and efficient tool for solving the single-source shortest path problem on a graph with non-negative edge weights.

3.2.2 Example

The example is from geeksforgeeks, a more detailed description can be found at here.



QUESTION: Given the graph shows above, and source = node(0); Find the shortest paths from the source to node(4).

Solution:
The minimum distance from node(0) to node(4) = 21. We need to take the following steps $0 \to 7 \to 6 \to 5 \to 4$. When using the Dijkstra's Algorithm to find the shortest path from node(0) to node(4), We use vector to keep track of the visited nodes. The vertices in order they are marked is as follow: $0, 1, 7, 6, 5, 2, 8, 3, 4$.

3.2.3 General Implementation by C++[15]

```
template<typename Location, typename Graph>
void dijkstra_search
  (Graph graph,
   Location start,
   Location goal,
   std::unordered_map<Location, Location>& came_from,
   std::unordered_map<Location, double>& cost_so_far)
{
  PriorityQueue<Location, double> frontier;
  frontier.put(start, 0);

  came_from[start] = start;
  cost_so_far[start] = 0;

  while (!frontier.empty()) {
    Location current = frontier.get();

    if (current == goal) {
      break;
    }

    for (Location next : graph.neighbors(current)) {
      double new_cost = cost_so_far[current] + graph.cost(current, next);
      if (cost_so_far.find(next) == cost_so_far.end()
          || new_cost < cost_so_far[next]) {
        cost_so_far[next] = new_cost;
        came_from[next] = current;
        frontier.put(next, new_cost);
      }
    }
  }
}
```

3.3 RRT Algorithm

RRT, or Rapidly-Exploring Random Tree, is an algorithm used for path planning and robot motion planning in complex environments. It is a probabilistic algorithm that generates a tree of possible paths by randomly exploring the environment and connecting these paths to a growing tree structure. As the tree grows, it becomes more likely to find a path to the goal, and the algorithm can quickly converge on an optimal solution.

3.3.1 Explanation

Rapidly-exploring Random Trees (RRT) is a algorithm used for path planning in robotics and other fields. The algorithm generates a tree of possible paths, starting from an initial location, and incrementally expands the tree by adding new random samples from the environment. The algorithm then uses the tree to find a path from the initial location to a goal location, by connecting the closest point in the tree to the goal. RRT are particularly useful in environments where the possible paths are not known in advance, or where the environment is highly dynamic. The algorithm is able to find paths quickly, even in large and complex environments, by using random sampling to explore the space. One of the key features of RRT is the use of a nearest-neighbor search algorithm to find the closest point in the tree to a new random sample. This allows the algorithm to efficiently connect new samples to the tree and avoid getting stuck in local minima. Overall, RRT are a useful algorithm for path planning in complex environments.

3.3.2 Example

The examples are from Tim Chinenov, it can be found at here.

3.3.3 Pseudo Code[16]

```
Rad = r
G(V,E) //Graph containing edges and vertices
For itr in range(0  n  )
    Xnew = RandomPosition()
    If Obstacle(Xnew) == True, try again
    Xnearest = Nearest(G(V,E),Xnew)
    Cost(Xnew) = Distance(Xnew,Xnearest)
    Xbest,Xneighbors = findNeighbors(G(V,E),Xnew,Rad)
    Link = Chain(Xnew,Xbest)
    For x' in Xneighbors
        If Cost(Xnew) + Distance(Xnew,x') < Cost(x')
            Cost(x') = Cost(Xnew)+Distance(Xnew,x')
            Parent(x') = Xnew
            G += {Xnew,x'}
    G += Link
Return G
```

4 Algorithms Analysis

4.1 Strengths for Each Algorithm

- **A* Search Algorithm**
  - A* is capable of finding the shortest path between two points, even in complex environments with many obstacles.
  - A* is flexible and can be easily adapted to different types of problems by using different heuristics, or rules of thumb, to guide the search process.
  - A* is efficient and can be used to solve large problems quickly, thanks to its use of heuristics to guide the search process and eliminate unnecessary steps.
  - A* is reliable and produces consistent results, as long as the heuristics used are consistent and accurate.
  - A* is easy to implement and understand, making it a popular choice for many applications.
- **Dijkstra's Algorithm**
  - Dijkstra's algorithm is capable of finding the shortest path between two points in a graph, even in complex environments with many obstacles.
  - Dijkstra's algorithm is efficient and can be used to solve large problems quickly, thanks to its use of a priority queue to prioritize the exploration of nodes.
  - Dijkstra's algorithm is reliable and produces consistent results, as long as all edge weights are non-negative.
  - Dijkstra's algorithm is easy to implement and understand, making it a popular choice for many applications.
  - Dijkstra's algorithm can be easily extended to handle additional constraints, such as time-dependent edge weights or multiple objectives.
- **RRT Algorithm**
  - RRT are capable of finding paths in complex environments with many obstacles, even in high-dimensional spaces.
  - RRT are probabilistic algorithms, which means that they can find a solution even when one is not guaranteed to exist.
  - RRT are efficient and can be used to solve large problems quickly, thanks to their incremental nature and the use of random sampling to guide the search process.
  - RRT are flexible and can be easily adapted to different types of problems by using different heuristics to guide the search process.
  - RRT are easy to implement and understand, making them a popular choice for many applications.

4.2 Limitations for Each Algorithm

- **A* Search Algorithm**
  - A* is only guaranteed to find the shortest path between two points if the heuristics used are consistent and admissible, meaning that they never underestimate the true cost of reaching the goal.
  - A* requires the use of heuristics, which can be difficult to come up with for some problems. In these cases, the performance of A* may be limited by the quality of the heuristics used.
  - A* can be computationally intensive, especially for large or complex problems. In these cases, it may be necessary to use more efficient variants of A* or to use other algorithms altogether.
  - A* is only applicable to problems that can be represented as a graph, where the nodes represent states and the edges represent the transitions between states. This may limit its applicability to some types of problems.
  - A* is not well-suited to dynamic environments where the costs of reaching the goal may change over time. In these cases, it may be necessary to use other algorithms that can adapt to changes in the environment.
- **Dijkstra's Algorithm**
  - Dijkstra's algorithm is only guaranteed to find the shortest path between two points if all edge weights are non-negative.
  - Dijkstra's algorithm is a greedy algorithm, which means that it only considers the cost of reaching the next node and does not consider the cost of reaching the goal from that node.
  - Dijkstra's algorithm is computationally intensive, especially for large or complex problems.
  - Dijkstra's algorithm is only applicable to problems that can be represented as a graph, where the nodes represent states and the edges represent the transitions between states.
  - Dijkstra's algorithm is not well-suited to dynamic environments where the costs of reaching the goal may change over time.
- **RRT Algorithm**
  - RRT are probabilistic algorithms, which means that they can find a solution even when one is not guaranteed to exist.
  - RRT can be computationally intensive, especially for large or complex problems.
  - RRT are only applicable to problems that can be represented as a graph, where the nodes represent states and the edges represent the transitions between states.
  - RRT are not well-suited to dynamic environments where the costs of reaching the goal may change over time.
  - RRT require the use of heuristics to guide the search process, which can be difficult to come up with for some problems.

4.3 Potential Future Research Directions

- Developing more efficient algorithms for path planning, such as algorithms that can solve large or complex problems more quickly or that can find solutions with higher quality.
- Investigating the use of machine learning and other artificial intelligence techniques for path planning, such as reinforcement learning, evolutionary algorithms, or neural networks.
- Exploring the use of path planning algorithms for new applications, such as autonomous vehicles, drones, or space exploration.
- Investigating the use of path planning algorithms in dynamic or uncertain environments, where the costs of reaching the goal may change over time or where the goal itself may be unknown or uncertain.
- Developing new heuristics or other methods for guiding the search process in path planning algorithms, such as methods that can incorporate additional information or constraints into the search process.

5 Summary

Path planning algorithms are a critical component of autonomous systems, such as robots and self-driving cars. These algorithms are used to generate feasible and safe paths for the system to follow, taking into account the system's capabilities and the characteristics of the environment in which it is operating. The algorithms are used to determine the optimal path for a vehicle or robotics to follow, the main goal is to find a path that avoids obstacles, reaches a destination efficiently, and satisfies other constraints such as traffic regulations.

In recent years, there has been significant research and development in the area of path planning algorithms. A variety of different algorithms have been proposed and tested, each with its own strengths and weaknesses. These algorithms can be broadly classified into two categories: local planning algorithms and global planning algorithms. Local planning algorithms are designed to generate paths in the immediate vicinity of the system, based on information about the immediate environment. These algorithms are typically fast and efficient, but they can only generate paths in a limited area and may not be able to find a feasible path if the environment is cluttered or has other constraints. Global planning algorithms, on the other hand, are designed to generate paths over a larger area, taking into account the global structure of the environment. These algorithms are typically more computationally intensive, but they are able to generate feasible paths in a wider range of environments and can find paths that avoid obstacles and constraints.

Overall, the field of path planning algorithms is an active area of research, with a wide range of algorithms being developed and tested. The choice of algorithm will depend on the specific application and the characteristics of the environment in which the system is operating. With the further development of hardware devices, sensors, and artificial intelligence, path planning will also see another peak in its development. I believe that in the near future, the application of path planning algorithms in the autonomous driving industry will bring the human society an unbelievable changes!

References

[1] Shneiderman, B. 2022. *Introduction: What are the goals of AI research?* Human-Centered AI, 87–92. https://doi.org/10.1093/oso/9780192845290.003.0011

[2] Frankenfield, Jake. "Artificial Intelligence: What It Is and How It Is Used." Investopedia, Investopedia, 12 Sept. 2022, www.investopedia.com/terms/a/artificial-intelligence-ai.asp

[3] Brown, Sara. "Machine Learning, Explained." MIT Sloan, 21 Apr. 2021, mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained.

[4] Staff, Editorial. "Ai Robots and Robotics: Difference, Current Examples of Artificially Intelligent Robots, and Role of Software Robots." Ai.nl, Editorial Staff Https://Secure.gravatar.com/Avatar/988c77cad41d1c55b988bff2c4600e8b?s=96&d=Mm&r=g,6Nov.2022,www.ai.nl/knowledge-base/robots-robotics/

[5] Banzhaf, Wolfgang, and Ting Hu. "Evolutionary Computation." Evolutionary Biology, 2019, doi:10.1093/obo/9780199941728-0122.

[6] Nakisa, Bahareh, et al. "Evolutionary Computation Algorithms for Feature Selection of EEG-Based Emotion Recognition Using Mobile Sensors." Expert Systems with Applications, vol. 93, 2018, pp. 143–155., doi:10.1016/j.eswa.2017.09.062.

[7] SAE. "The 6 Levels of Vehicle Autonomy Explained." Synopsys Automotive, www.synopsys.com/automotive/autonomous-driving-levels.html

[8] Mochihashi, D. (2020). Robotics, grounding and natural language processing. Journal of Natural Language Processing, 27(4), 963–968. https://doi.org/10.5715/jnlp.27.963

[9] Ikeuchi, K., Matsushita, Y., Sagawa, R., Kawasaki, H., Mukaigawa, Y., Furukawa, R., amp; Miyazaki, D. (2020). Robot Vision, autonomous vehicles, and Human Robot Interaction. Active Lighting and Its Application for Computer Vision, 289–303. https://doi.org/10.1007/978-3-030-56577-0_12

[10] Shi, W., amp; Liu, L. (2021). Autonomous driving landscape. Computing Systems for Autonomous Driving, 1–18. https://doi.org/10.1007/978-3-030-81564-6_1

[11] Budrich, V. B. (2020). Defending autonomous driving. Keeping Autonomous Driving Alive, 103–118. https://doi.org/10.2307/j.ctv10kmf41.10

[12] Cadence System Analysis. "The Use of Radar Technology in Autonomous Vehicles." Cadence, 13 Oct. 2022, https://resources.system-analysis.cadence.com/blog/msa2022-the-use-of-radar-technology-in-autonomous-vehicles

[13] A* search algorithm. GeeksforGeeks. (2022, May 30). Retrieved December 10, 2022, from https://www.geeksforgeeks.org/a-search-algorithm/

[14] Prasanna. (n.d.). A* algorithm (graph traversal and path search algorithm). enjoyalgorithms. Retrieved December 12, 2022, from https://www.enjoyalgorithms.com/blog/a-star-search-algorithm

[15] Dijsktra's algorithm. GeeksforGeeks. (2022, August 31). Retrieved December 10, 2022, from https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/

[16] Abiy, T., Pang, H., amp; Tiliksew, B. (n.d.). A* search. Brilliant Math amp; Science Wiki. Retrieved December 10, 2022, from https://brilliant.org/wiki/a-star-search/#:~:text=The%20pseudocode%20for%20the%20A,presented%20with%20Python%2Dlike%20syntax.&text=The%20time%20complexity%20of%20A,search%20space%20is%20a%20tree