**Proposal**

**Student name:**

**Yaobin Liang**

**Advisor:**

**David Paul Benjamin**

**Date: 9/21/2022**

Table of Contents

[1.1 Background and Basic Concept 5](#_Toc114675344)

[1.2 Piano Mover’s Problem 6](#_Toc114675345)

[1.3 Configuration Space 9](#_Toc114675346)

[1.4 Degrees of Freedom 10](#_Toc114675347)

[1.5 Robotic Arm 10](#_Toc114675348)

[1.5.1 Redundant DOF for Manipulation Robot 12](#_Toc114675349)

[1.5.2 Link, Joint and End-effector 13](#_Toc114675350)

[1.5.3 Joint Limit 14](#_Toc114675351)

[1.5.4 Task Space and Workspace 15](#_Toc114675352)

[1.5.5 Forward Kinematics and Inverse Kinematics 18](#_Toc114675353)

[1.5.6 Motion Planning for Robotic Manipulators 21](#_Toc114675354)

[1.5.7 Low-dimensional vs High-dimensional in Static Environment 22](#_Toc114675355)

[1.5.8 Motivation 22](#_Toc114675356)

[1.5.9 Classical Problem Formulation 23](#_Toc114675357)

[1.5.10 Research Problems in Motion Planning 23](#_Toc114675358)

[2.1 Related Work in Motion Planning for Manipulators 25](#_Toc114675359)

[2.2 Grid-based 27](#_Toc114675360)

[2.3 Sampling Based 29](#_Toc114675361)

[2.4 Geometric Method Based 35](#_Toc114675362)

[2.5 Machine Learning Based 37](#_Toc114675363)

[2.5.1 Supervised Learning Based 37](#_Toc114675364)

[2.5.2 Unsupervised Learning Based 39](#_Toc114675365)

[2.5.3 Reinforcement Learning Based 40](#_Toc114675366)

[3.1 Approach of This Study 41](#_Toc114675367)

[3.1.1 Limitation of existing supervised learning methods 42](#_Toc114675368)

[3.2 Denavit–Hartenberg Parameters 46](#_Toc114675369)

[3.3 Moment of Inertia 47](#_Toc114675370)

[3.4 Simulated Environment 49](#_Toc114675371)

[3.5 Data Acquisition/Data Preparation 51](#_Toc114675372)

[3.5.1 Static case 54](#_Toc114675373)

[3.5.2 Dynamic case 55](#_Toc114675374)

[3.6 Deep Learning Structure 56](#_Toc114675375)

[3.7 Evaluation 58](#_Toc114675376)

[3.8 Other Hardware and Software Maybe Used 58](#_Toc114675377)

[3.9 Novelty of This Research: 58](#_Toc114675378)

[Reference 59](#_Toc114675379)

[Appendix 69](#_Toc114675380)

[Figure 1. The piano mover’s problem. The Green line is the collision-free path generate by RRT-Connect algorithm. 7](#_Toc114674940)

[Figure 2. The full trajectory of Figure 1. 8](#_Toc114674941)

[Figure 3. Kinova Gen3 7-DOF robot with the vision module 9](#_Toc114674942)

[Figure 4. Four commonly used robot arms. Upper left: the Franka panda robot, upper right: the Kinova Gen3 robot with gripper, bottom left: the UR10e, bottom right: the KUKA LBR IIWA 7 R800. 12](#_Toc114674943)

[Figure 5 . Two types of 2D joints: a revolute joint and a prismatic joint 13](#_Toc114674944)

[Figure 6. A schematic of a 2D manipulator with three revolute joints 14](#_Toc114674945)

[Figure 7. (a) A 2\_link planar robot arm in 2D, l1 = l2 = 10 cm, θ1 ∈ [ 0, 2/3π], θ2 ∈ [ 0, π] (b) The workspace of a 2\_link robot with constraints on joint angles. 16](#_Toc114674946)

[Figure 8. Side view of Kinova Gen3 7 Dof robot nominal reachable workspace (measurements in mm) 16](#_Toc114674947)

[Figure 9. 3D view of the Kinova Gen3 7 DOF robot reachable workspace. 17](#_Toc114674948)

[Figure 10. The schematic representation of forward and inverse kinematics. 18](#_Toc114674949)

[Figure 11. Two distinct joint configurations: the black color links is one possible combination and the gray color links is the other possible combination. 20](#_Toc114674950)

[Figure 12. Lorenz attractor. 26](#_Toc114674951)

[Figure 13. Left: the two-link, planar-arm robot with two circular obstacles in the 2D plane. θ1 and θ1 are the joint angles. θ1 ∈ [ 0, 2π), θ2 ∈ [ 0, 2π). Right: the configuration space of the two-link robot. The red-colored area is the infeasible configuration space formed by the obstacles. The gray area is the available space that the arm can move. The green line is the solution generated by BFS that can connect the initial and final configuration. 29](#_Toc114674952)

[Figure 14. In a 2D environment, the starting position is the center at (0,0), and the green point is the goal position. A rapidly exploring random tree (RRT) grows a tree rooted at a start node. The green line along the edges is the solution found by the RRT algorithm. 30](#_Toc114674953)

[Figure 15. The green line is the path generated by RRT\* algorithm with the same starting and goal positions shown in Figure 12. 31](#_Toc114674954)

[Figure 16. The PRM in 2D environment with 100 nodes. 33](#_Toc114674955)

[Figure 17. The starting point is (2,2) and the goal point is (12,11). The orange line is the path generated by PRM algorithm. 34](#_Toc114674956)

[Figure 18. The roadmap has 20 sample nodes. The yellow line is the path generated by PRM algorithm. The end point of Kinova Gen3 robot is going to move from target1 to target2, and the robot will follow the yellow path. 35](#_Toc114674957)

[Figure 19. The 3-joints robot in the artificial potential field. The goal will attract the robot while the obstacle will repel the robot. 36](#_Toc114674958)

[Figure 20. Figure 20. One Simulated Environment in Klampt. There are ten obstacles, two moving spheres in green and eight static spheres in gray. The Kinova Gen3 with vision module robot is sitting at (0, 0, 0) position with a random initial configuration. The brown color rectangle rod and the ground are the terrain models. 50](#_Toc114674959)

[Figure 21. Render the environment in Klampt with the robot’s movement trajectory. 54](#_Toc114674960)

[Table 1. The Kinova Gen3 7 DOF spherical classical DH parameters. 44](#_Toc109775048)

[Table 2. Moments of inertia of the Kinova Gen3 7 DOF robot with vision module. 46](#_Toc109775049)

[Table 3. Example obstacles information in environment#10 48](#_Toc109775050)

[Table 4. Sample trajectory of environment # 25, initial and goal configurations were randomly generated. A Kinova gen3 robot model was used to generate trajectories. 50](#_Toc109775051)

**High-dimensional Spaces Motion Planning for Robotic Arm in Dynamic Environment**

This study is about solving the motion planning problem in the dynamic environment with high-dimensional redundant manipulators robotics. The concept of a dynamic environment refers to an environment involving static and moving obstacles, as well as obstacles whose position at any given time can be known, but not their trajectory, due to internal or external uncertainties. Finally, a deep neural network architecture that considers the robotic arm mechanism and its shape will be designed and tested in a simulated dynamic environment using the Kinova Gen3 robots and vision modules.

1.1 Background and Basic Concept

Robotics is the combination of mechanical structure and built-in function. This includes the capability for computing, mapping, sensing, actuation, and so on. Those functions use machines (robots) to perform tasks done traditionally by humans. However, robots do not automatically and intelligently carry out these tasks. The fundamental need in this area is algorithms that can connect the high-level specifications of tasks from humans and the low-level machine actuation. For instance, given the high-level command of moving the robotic from point A to point B. The algorithm should be able to guide and send out signals about how much current will run through the electric device so that the rotors can precisely perform rotations. The term of motion planning is often used for these kinds of problems.

1.2 Piano Mover’s Problem

The Piano Mover’s Problem [33] is the classical motion planning problem. Given a three-dimensional (3D) environment and a set of 3D rigid body objects, the problem is to find a free-fly path from a starting pose to a goal pose without colliding with the rigid body objects. Here, the pose includes both the position and orientation information of a coordinate frame. The 3D rigid body objects are also called obstacles, and they are assumed to be static; the boundary of the environment is fixed. If all the above information passes to the algorithm as input, then it will output a collision-free path that can connect the start and goal pose. Figure 1 shows the path that was generated from RRT-Connect [18] planning algorithms for the Piano Mover’s Problem. Figure 2 is the trajectory representation of the solution that was generated by RRT-Connect planning algorithms in Figure 1. Executing the planned path should let the piano move to the goal pose without colliding with other obstacles. The other variation on this problem is the generalized mover’ problem, where the robot may consist of several links and joints, e.g., an articulated robot. Figure 3 shows the Kinova Gen3 [12] robot, and it is an articulated robot.

Figure . The piano mover’s problem. The Green line is the collision-free path generate by RRT-Connect algorithm.

Figure . The full trajectory of Figure 1.



Figure . Kinova Gen3 7-DOF robot with the vision module

1.3 Configuration Space

In the Piano Mover’s Problem, the goal is to move the pinao from the initial location to a desired location. The piano can be considered as the free-flying robot. The key problem is to make sure there is no collision between the robot (piano) and the obstacles. Due to the fact of how robots move, the robots have their own way of representing the location of all the points on the robot. The term configuration, which is the complete specification of the location of every point on the robot, will be used in robotics. The configuration space (C-Space) [34, 35, 36] is the space in which the robot can make on all possible configurations. For example, in Figure 1, the goal is to move the robot (piano) from configuration q0= (x0, y0, z0, α0 ,β0 ,γ0 ) to q1= (x1, y1, z1, α1, β1, γ1 ) without colliding with other obstacles, where xi, yi, zi is the Cartesian distance, αi,βi,γi is the orientation angle respect to the coordinate axes, i = {0,1}. Figure 2 depicts the sequence of configurations that can connect the initial and goal configurations.

1.4 Degrees of Freedom

The robots’ workspace (or workspace envelope) is the set of all points the robots can reach in the physical embedding volume. It’s easy to represent the robot's position in the workspace. However, it’s different while representing the robot in C-Space. In general, C-Space is not Euclidean space. So, it does not look like the n-dimensional Euclidean space n. The dimension of the C-Space is the independent variable in the representation of the configuration. The term degrees-of-freedom (DOF) is used to describe the number of independent dimensions of the C-Space for the robot.

In the Piano Mover’s Problem, the piano has six degrees of freedom: three to represent the Cartesian coordinate location (x, y, z) and three to represent the orientation (roll, pitch, yaw).

1.5 Robotic Arm

Robotic arms are programmable robot manipulators with similar functions to human arms. The robotic manipulators have been used in factory automation to engage in repetitive and boring tasks, such as handling radioactive substances and working in dangerous or messy environments. Figure 4 shows four commonly used robot arms. The upper left is the Franka panda robot; the upper right is the Kinova Gen3 robot with gripper; the bottom left is the UR10e; the bottom right is the KUKA LBR IIWA 7 R800. These kinds of robots are generally motor driven (servo). The most common use of motor-driven robotics is for performing continuous, rapid, and repeatable tasks for long periods of time, and they are especially useful for industries such as manufacturing, machining, and assembly.

The robot arms typically consist of a series of joint mechanisms, articulations, and manipulators that closely emulate the motion and functionality of a human arm.

Figure . Four commonly used robot arms. Upper left: the Franka panda robot, upper right: the Kinova Gen3 robot with gripper, bottom left: the UR10e, bottom right: the KUKA LBR IIWA 7 R800.

1.5.1 Redundant DOF for Manipulation Robot

It is common for manipulators that are working in spatial 3D environment to be able to be positioned and oriented in space arbitrarily; and they are required to have 6 degrees of freedom. The manipulators that have more DOF than they need will be considered redundant DOF.

An example of a kinematically redundant manipulator would be one that has more than six degrees of freedom for repositioning and orienting its end-effector in space. With extra degrees of freedom, a manipulator can have greater control over motion and be more dexterous and flexible.

1.5.2 Link, Joint and End-effector

The robotic arms eventually must work for humans to perform some tasks. Such as picking up heavy objects. In the process of picking up objects, a mechanical system that has the main purpose of transferring motion and forces from one to another output is very important. The robot is mechanically connected by a set of rigid bodies called links. And those links are connected by joints. The joints define the relationship between adjacent links. The joints can be pin joints or sliding joints, so that joints allow either rotational motion or translational displacement. Figure 5 shows two types of 2D joints: the revolute joint and the prismatic joint. The revolute joints allow one link to rotate with respect to the other connected link, while the prismatic joint allows one link to translate with respect to the other connected link. Diagram

Description automatically generated with medium confidence

Figure . Two types of 2D joints: a revolute joint and a prismatic joint

The actuators are the mechanical parts of the robot that deliver forces or torques to the joints so that the robot’s links can move. Such as the electric motor or servo. The end-effector, such as the sprayer to perform the painting tasks or the gripper to grasp the objects, is attached to the end of the robotic arm. Figure 6 is a 2D example of the robotic arm schematic with respect to the links, joints, and end-effector.

Diagram

Description automatically generated

Figure . A schematic of a 2D manipulator with three revolute joints

1.5.3 Joint Limit

The joint limits are values that define the mechanical parts of motion allowed to rotate or translate. For example, the interval of joint values [0, π] is the joint limit of a revolute joint. Some revolute joints may have no limit, such as a motor that drives a drill as the end-effector. Those joints without a limit are known as continuous rotation joints.

1.5.4 Task Space and Workspace

The robot's task space is the natural environment in which the robot's task is expressed. If the task of the robotic arm is to move the end-effector to a specific point and pose, then the task space is the 6-dimensional space of a rigid body configuration. Whereas the task is only to move the robotic arm to a specific location and do not care about the pose, then the task space is the spatial 3D Euclidean space.

The workspace is the physical volume of the cartesian points that the end-effector can reach. Figure 7 (a) is a two links articulated robot in the 2D plane. The length of link1 and link2 are equal to 10 cm, denote as l1 = l2 = 10cm. The first joint of the robot denotes as θ1 and the second joint is θ2, where θ1 ∈ [ 0, 2/3π], θ2 ∈ [ 0, π]. Figure 7 (b) shows the workspace of the 2\_link robot in 2D plane. The green region bounded by the blue line is the workspace for the 2\_link robot. Figure 8 is the side view of the real-world robot Kinova Gen3 7DOF robot’s workspace. And Figure 9 is the 3D view of the Kinova Gen3 7 DOF robot’s reachable workspace.

Chart, line chart

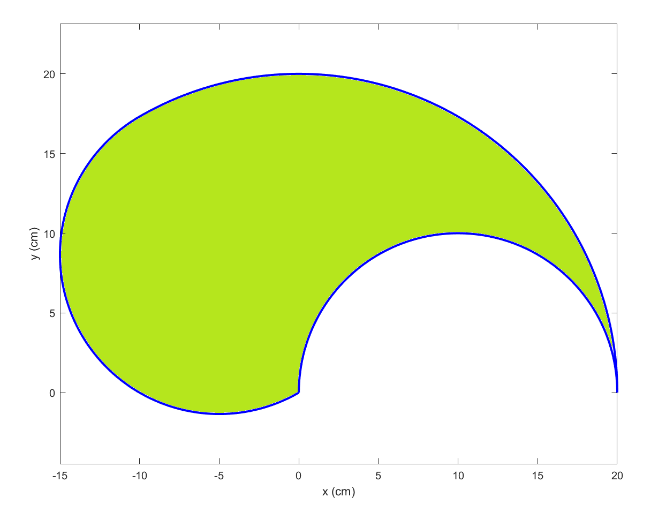
Description automatically generated

Figure . (a) A 2\_link planar robot arm in 2D, l1 = l2 = 10 cm, θ1 ∈ [ 0, 2/3π], θ2 ∈ [ 0, π] (b) The workspace of a 2\_link robot with constraints on joint angles.

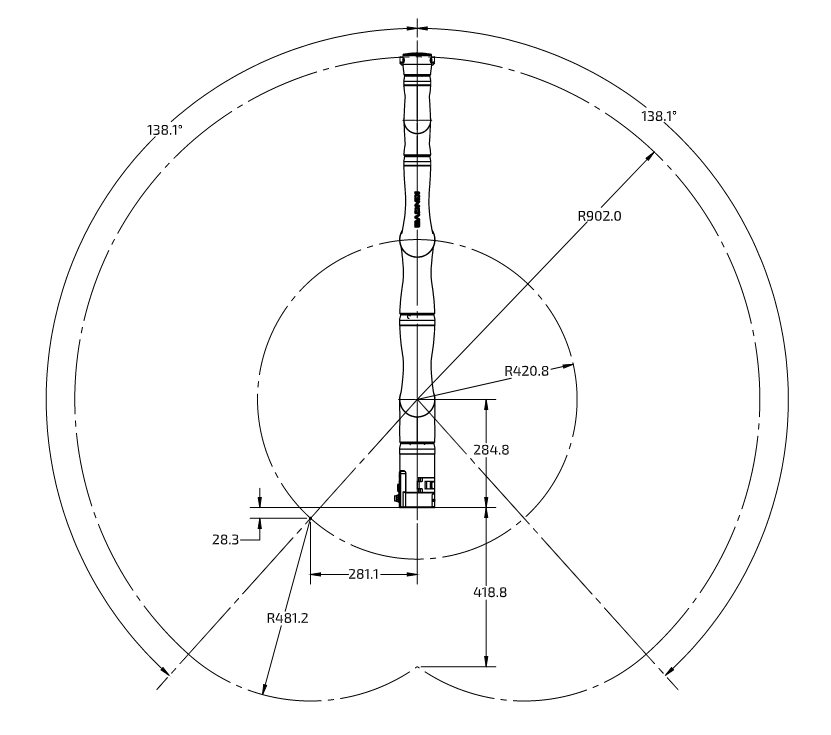


Figure . Side view of Kinova Gen3 7 Dof robot nominal reachable workspace (measurements in mm)



Figure . 3D view of the Kinova Gen3 7 DOF robot reachable workspace.

Free Space defines where a robot can move. The robot may not be able to move around certain areas of the workspace, for example due to obstacles. If the robot cannot move in these spaces, the planning algorithm such as the rapidly exploring random tree (RRT) [17] will remove them from the configuration space.

1.5.5 Forward Kinematics and Inverse Kinematics

Forward kinematics is about finding the position and orientation of the end point of the robotic arm given the joint angles along the multi-joint arm robot and other constants. Here, the constants refer to the size of the robot arms, such as the length, width of the links. This forward kinematics process is about going forward from the base of the robot towards the end point of the robot’s arm. The inverse kinematics (IK) are essentially the complete opposite of the forward kinematics. The inverse kinematics [41] is given the position and orientation of the end-effector and other constants, and then finds the joint variables of the multi-joint arm robot. The relationship between forward kinematics and inverse kinematics is illustrated in Figure 10.

Figure . The schematic representation of forward and inverse kinematics.

Solving the forward kinematics problem is easy but solving the inverse kinematics problem is hard. The forward kinematics calculation is straightforward, you just need to plug into the function, and there is a unique solution output. However, the solution may be infeasible due to collisions between the robot links themselves or with other obstacles. When calculating the inverse kinematics problems, there are often more than one solution and numerous approaches to finding the solution.

Now, let’s consider Figure 7 (a) as one example for solving both forward and inverse kinematics. Suppose we need to find the location of end point of the robot in Figure 7 (a). Given θ1 is the angle between link1 and the x-axis, θ2 is the angle between link1 and link2. The length of link1 and link2 are the constant numbers l1 and l2. Using forward kinematics method to find the position information of the robot:

Or

In contrast, if the inverse kinematics method is utilized to calculate the joint angles θ1 and θ2 with given end point position X= (x, y) and length of link1 & 2:

There may existing 2 distinct joint configurations q ∈ 2 for a given end point position X ∈ 2 as shown in Figure 11. The two different sets of configurations can be written as:

and

Figure . Two distinct joint configurations: the black color links is one possible combination and the gray color links is the other possible combination.

1.5.6 Motion Planning for Robotic Manipulators

Motion planning is important in the field of manipulator robotics. Motion planning is to find a way that how to manipulate a robot to reach the desired configuration without collisions. One of the most important factors in motion planning is collision detection. The Planners need to ensure that the robot will move without colliding with the robot’s links themselves and the other obstacles. Otherwise, the robotic system will stop working due to mechanical crash. An automatic motion planner or human program must be used to do this.

Suppose we want a welding robot to weld a thin seam on an object. The welding robot needs to know how to configure itself so that the destination is reached while avoiding obstacles such as its partners (other robots) and/or the welding robot itself. The robot must be able to maneuver itself appropriately to follow a range of motions throughout the welding process. So, a series of waypoints or configurations must be planned before executing the welding process.

Also, when the robot is in a complex environment, motion planners are useful because they speed up the programming process. Instead of planning every step manually, the motion planner can generate one or more routes that the robot can follow automatically.

To summarize, motion planning is very important for the robotic manipulator. By using the motion planner, people don't have to plan every single movement manually for the robot. The motion planner will automatically create one or more collision-free routes for the robot to follow.

1.5.7 Low-dimensional vs High-dimensional in Static Environment

We can consider it a low-dimensional problem when the DOF of the robot is less than 3. Point robots that move in 2D environment with static polygonal obstacles can be considered as the low-dimensional problem. Many algorithms, such as the Bug algorithms [37, 38] and the grid search algorithms, can handle the low-dimensional motion planning problem efficiently.

When the DOF of the robot is greater or equal to 3, the problem will become more complex. Such as moving the quadcopters that avoid obstacles in 3D, manipulating the n-joints articulated robot arms, collaborating with multi-robot in a team. In low-dimensional cases, the main issue is how to represent the obstacles in the configuration space. However, in higher-dimensional configuration space, the main issue is different. It is fairly fast to build a 2D grid and perform search algorithms in low-dimensional cases. It will quickly return the path if it exists. However, as the number of DOF increases, running time and memory consumption exponentially grow. It is impractical to perform grid-search or bug algorithms for higher-dimensional cases.

1.5.8 Motivation

Most of the existing motion planning algorithms are typically designed for robots that work in the static environments, and because of this, most robot manipulators are restricted to the predetermined motion trajectories, and the robots’ responses are non-reactive. It would be beneficial to increase the reactivity of robot manipulators. Robots could move without knowing a pre-built model of the environment; robots could move less depending on precalculated paths and plan their motion or path on the fly, reacting to unexpected events such as moving obstacles.

1.5.9 Classical Problem Formulation

In this section, the classical motion planning problems is formalized.

Let denote the configuration space. Where and d represents the problem dimension. The obstacle set denoted as . The free configuration space set denoted as . and represent the initial configuration and goal configuration. T represent the time.

The aim of motion planning is to find the collision-free path such that and.

1.5.10 Research Problems in Motion Planning

Avoiding collisions with obstacles is very important for manipulators. Otherwise, the manipulator or held object may cause serious damage. The increased complexity is due to the greater chance of robots colliding with themselves and other obstacles. Motion planning in real time is important for the motion of articulated robots, since it permits different parts of the robot to move simultaneously, avoiding collisions with each other.

Motion planning is challenging for dynamic systems with moving obstacles, particularly when they have dynamic constraints and require computing solutions fast and accurate enough to be useful for real-time implementation.

The dramatic increase in time and space complexity as the number of DOF increases has led to the development of several approximate methods that fit in high-dimensional cases. The main revolution algorithms such as sampling-based algorithms RRT and probabilistic roadmaps (PRM) [26], were developed for the high dimensional space motion planning problem. The sampling-based and PRM algorithms are probabilistic complete but not optimal solutions for the motion planning problem. Soon after, the researchers proposed the probabilistic complete and asymptotic optimal solution for motion planning problems such as RRT\* [20], PRM\* [20]. The main idea is to rewire the new node to the nearest node. However, those algorithms were relatively slow due to the complexity of time and space in dynamic environments. As the it requires rapid computing and output the path all the time in the dynamic environment. Since the position of obstacles is changing, it may hit the robot during the execution of the calculated motion plan.

Planning a feasible path in high-dimensional spaces from the start configuration to goal configuration in the static environment for a robot is easier as compared to planning in the dynamic environment where the position of obstacles may change as a function of time. As a matter of fact, the piano mover’s problem, which is the basic version of the motion planning problem that in the static environment is proven to be PSPACE-hard [63, 75]. As the superset of the basic motion planning problem, planning in dynamic environments is even harder. At the moment of writing this proposal, no software can perform efficient motion planning tasks for high-dimensional spaces in a dynamic environment. Finding a feasible path for the high DOF robot in the dynamic environment is still a challenging task, and there is a need to develop such an effective technique for motion planning in the dynamic environment with the high DOF robot. That is the goal of this research.

2.1 Related Work in Motion Planning for Manipulators

Motion planning is one of the fundamental problems in the robotic area [1, 2]. Motion planning can be defined as finding a feasible collision-free path from the initial configuration to the goal configuration [1, 2]. Many algorithms have been developed to solve this basic problem [1, 2, 3]. The existing popular methods are the grid search method, the sampling-based method, the geometric based method, and the artificial potential field planning method [2, 3]. However, the above methods cannot solve the problem effectively in high-dimensional space with dynamic environments [4]. A dynamic environment is defined as a workspace in which obstacles are still and moving, and where the position of obstacles at any given time can be determined.

There are two different approaches for motion planning in high-dimensional space with dynamic environments. The first approach is to observe the environment for a while and then output a feasible collision-free path. This kind of approach can be expensive as you may not know if the environment is changing periodically or not. If the motion of the obstacles is nonperiodic motion, the observation stage will be infinitely long. The Lorenz system [50] is one example of nonperiodic motion. Figure 12 shows the trajectory of a point's motion for the Lorenz system. If we replace the point with the obstacles, that would be the nonperiodic motion example. This observation time is infinite, and that impractical for the planner to output a collision-free path for the robot.

Figure . Lorenz attractor.

The observation in these kinds of methods is to get the motion pattern of the obstacles. And the planner can find a feasible solution according to the observed pattern of motion of all obstacles. These kinds of methods may not be the best choice for an environment that requires immediate action from the robot.

The second approach is different, and the observation period is not required. The second approach is to output the feasible solution at all times as the robot moves. These kinds of methods will consider the environment as a static environment at every moment, and the planners can output the feasible solution as the program sends out a request. These kinds of methods require that planners be fast enough to output the feasible collision-free path. Otherwise, the robot may collide with the obstacles as the robot followed the old collision-free path at time t, but the obstacles had already moved to the other position at time t+1.

The following paragraphs will present various robotic algorithms to solve motion planning problems for articulated robots.

2.2 Grid-based

The grid-based method will overlay the grid on the configuration space, and each configuration is considered a grid point. The robot can move from one grid point to another grid point if it contains an obstacle-free path. Breadth-first search (BFS), depth-first search (DFS) are the based search algorithms. These two search algorithms will work in low dimensional space only. As the number of degrees of freedom increases, the amount of memory required to represent the grid, as well as the time to search the graph, increases rapidly. In Figure 13, a breadth-first search is used to find the shortest path between an initial and final configuration of a two-link, planar-arm robot. First, divide each of the n dimensions into k intervals, and create kn grid cells; in this example, n = 2, and k ∈ 𝕫. Each cell represents the corresponding configuration of the robot. And then, perform the BFS algorithm to find the shortest path that connects the starting configuration and the goal configuration.

The search algorithm such as Dijkstra's [16] and A\* [15] will be used to find an optimal path that connects the start to the goal configuration. The A\* [15] algorithm and its variants [13, 25] are also commonly used in applications, such as Google Maps and other traffic systems. These algorithms can find the shortest path in the static environment if it exists. But A\* and its variants are not available in dynamic environments. In order to support dynamic environments, D\* [14], D\* lite and its variants were developed. However, D\* [14] and its variants [21, 22, 65] do not guarantee solution quality in a large dynamic environment. Overall, increasing the number of degrees of freedom increases the memory requirement, and the amount of time it takes to search the graph. This will lead to failure when grid search algorithms apply to high-dimensional search problems. The system halted due to the huge consumption of memory or search time. So, the major drawback of grid-based methods is that this approach is not practical for high-dimensional space problems due to the consumption of time and memory.

Figure . Left: the two-link, planar-arm robot with two circular obstacles in the 2D plane. θ1 and θ1 are the joint angles. θ1 ∈ [ 0, 2π), θ2 ∈ [ 0, 2π). Right: the configuration space of the two-link robot. The red-colored area is the infeasible configuration space formed by the obstacles. The gray area is the available space that the arm can move. The green line is the solution generated by BFS that can connect the initial and final configuration.

2.3 Sampling Based

Rapidly-exploring random tree (RRT) [17] and its variants [18, 19, 20, 40, 42, 43, 48, 49, 64] are one of the most popular sampling-based algorithms in motion planning. These methods take samples from the obstacle-free configuration space and then connect all the sample nodes. Finally, it finds a path from the start configuration to the goal configuration by using the graph search algorithm. RRT and its variants can find the solution fast in high-dimensional space if the solution exists. Figure 14 and Figure 15 are examples of using RRT and RRT\*. In the RRT algorithm, first a goal point is selected, then edges are added from the closest node within the tree toward the goal point. RRT\* [20] improves on this by restructuring the tree so that the shortest paths are formed. The RRT\* is the asymptotically optimal algorithm with time complexity of O(logn) under the assumption of O (1) for collision check [20]. It is extremely helpful in finding an optimal path when faced with obstacles in dense fields that are full of obstacles. RRTs are designed to explore high-dimensional spaces efficiently. However, RRT and its variants couldn’t handle the high-dimensional problem in dynamic environments because they were slow compared to the learning method.

Figure . In a 2D environment, the starting position is the center at (0,0), and the green point is the goal position. A rapidly exploring random tree (RRT) grows a tree rooted at a start node. The green line along the edges is the solution found by the RRT algorithm.

Figure . The green line is the path generated by RRT\* algorithm with the same starting and goal positions shown in Figure 12.

The probabilistic roadmap (PRM) [26] and its variants [20, 27, 47] planners work fine when starting or goal configurations change but keep the environment unchanged. The PRM method will construct the roadmap first and then perform the graph search on the built roadmap. This method was designed to solve the multiple query problem. PRM and its variants will re-use and expand the previously built roadmap. And then it performs the search algorithm on the same roadmap with different start or goal configurations. PRM\* [20] is a batch variable-radius PRM. The PRM\* achieves asymptotic optimality and computational efficiency in multiple-query problems by scaling the radius with the number of samples.

Figure 16 and Figure 17 are examples of PRM in a 2D environment. First, the PRM algorithm generated some random points within the robot’s free workspace and then attempted to connect these points to develop a collision-free roadmap in the workspace. This step is shown in Figure 16. In Figure 16, it generates 100 nodes (points) in the 2D environment and then connects the nodes to build a roadmap. Secondly, the planner uses the map generated in the previous step to find the shortest collision-free path connecting the two target points. This step is shown in Figure 17. Figure 17 shows the feasible path from the starting points to the goal point. Increasing the number of nodes can increase the efficiency of the path by providing more feasible routes. On the other hand, increasing the number of nodes also increases computation time and complexity. To get good coverage of the map, you might need many nodes. As nodes are randomly placed on the map, some areas may not have enough nodes to connect to the rest of the map. Figure 18 shows how the PRM algorithm works for real-world robots. First, the PRM algorithm generates 20 random nodes in the workspace, and then it forms the roadmap by connecting all the nodes. Finally, the search algorithm will be used to find the shortest path (the yellow line in Figure 18) that connects the two target points, as shown in Figure 18. These PRMs methods may not work in dynamic environments because the environment is changing, and it cannot build a stable roadmap.

Figure . The PRM in 2D environment with 100 nodes.

Figure . The starting point is (2,2) and the goal point is (12,11). The orange line is the path generated by PRM algorithm.

Figure . The roadmap has 20 sample nodes. The yellow line is the path generated by PRM algorithm. The end point of Kinova Gen3 robot is going to move from target1 to target2, and the robot will follow the yellow path.

2.4 Geometric Method Based

The Geometric method such as Visibility graph [23], Voronoi diagrams [24] work great in low dimensional path planning problems. These types of algorithms, however, tends to be difficult to implement in higher dimensions involving complex constraints.

The artificial potential field’s (APF) methods [44, 45, 46, 53, 54, 55, 56, 57, 58] are another geometric approach. Every point in the workspace is assigned a potential field based on the potential field functions. The artificial potential field has potential values assigned to each position that determine its energy level. And the starting point will have the greatest potential, while the goal point has the lowest. By inserting a robot or planning agent into the artificial potential field, it moves toward decreasing the potential field. Figure 19 shows how the manipulator works in the artificial potential field. The blue circle is an obstacle. It will repel the robot. The red circle is the goal position, and it will attract the robot.

Figure . The 3-joints robot in the artificial potential field. The goal will attract the robot while the obstacle will repel the robot.

In higher dimensions with complex constraints, however, the APF algorithms tend to be computationally intractable. So, these APF based methods will also not work in high dimensional space or dynamic environment.

2.5 Machine Learning Based

The computational capability of motion planning, particularly for systems with dynamics, is hindered by the time required to compute solutions to complex constraints and obtain them fast enough to be used in real-time dynamic environment. The machine learning method can handle these kinds of problem very well.

In recent years, many machine learning based algorithms have been developed to solve motion planning problems. To guide the robot's movement, they use either the pre-designed reward function or learn from previously successful planning experiences. In general, machine learning based methodologies applied to robot motion planning can be classified as supervised, unsupervised, or reinforcement learning.

The following paragraphs will present the existing supervised, unsupervised, and reinforcement learning based motion planning methods for robotic manipulators, respectively.

2.5.1 Supervised Learning Based

In general, the supervised learning algorithms teach the system to classify data or predict outcomes based on training data. Suppose we have the input variables(X) and output variable (Y). Supervised learning is to learn the mapping function from input to output:

Essentially the supervised learning algorithms aiming to approximate the mapping function such that, when you have new input data X, you can predict the output variables Y. A vector is typically used to represent a set of inputs, with different dimensions representing the different attributes or features. The output is the vector too, however each dimension can be the continuous real numbers or the class label.

Qureshi et al. [5] proposed a Deep Neural Network (DNN) based iterative motion planning structure called "Motion-Planning Networks" to evaluate multiple motion planning scenarios, including the 7 DOF redundant manipulator. Their architecture consists of an encoder network and a planning network. The point cloud depicting the surroundings obstacles in the environment is encoded via the encoder network and fed into a planning system for generating predicted the collision-free paths from the initial configuration to the target configuration. Their results show that their proposed structure can output the feasible trajectories within 1s with average success rate of 85%. After that, Qureshi et al. [51] expanded the previous work in more challenging environments and demonstrated better performance metrics for MPNet. In [52], Bency et al. developed the OracleNet, a Recurrent Neural Networks (RNN) based motion planning algorithm that can generate near-optimal collision-free trajectories paths for static environments iteratively.

In paper [59], they propose the latent sampling-based motion planning network structure, which can output the motion plans for complex robot system by learning a plannable latent representation. A learned latent space is constructed via auto-encoding, dynamic, and collision checking networks, which could be regarded as sampling, local connecting, as well as collision checking for classical sampling-based motion planning. In [60], the generative adversarial network (GAN) [76] based model called Causal InfoGAN is presented to guide the robot move from initial configuration to goal configuration. In [66], they present Constrained Motion Planning Networks X, which is based on a conditional deep neural generator and discriminator with a neural gradients-based fast projection operator. All the above supervised learning algorithms can learn from the input data and can generate a sequence of configuration that connect the initial and goal configurations. However, the existing works did not consider the speed information of the robot and obstacles, which may not be suitable for manipulators’ motion planning in complex dynamic environments with moving obstacles.

2.5.2 Unsupervised Learning Based

Data that has not been labeled is modeled by unsupervised learning, often by grouping similar things together. Clustering, topic modeling, and latent semantic analysis [65] are other terms for unsupervised learning.

Compared with motion planning algorithms based on supervised learning, methods based on unsupervised learning are rarely used for motion planning problems [65]. We cannot find too many unsupervised learning based algorithms for motion planning problems. So, we won’t cover too much in this section.

In [67], inspired by reinforcement learning, an unsupervised learning path planning algorithm, Plan2vec, is proposed. In Plan2vec, a weighted graph is constructed using adjacent distances in the image dataset, and this measure is then extrapolated to the global embedding by extracting path integrals over the planned paths. The result of their study reveals that it can significantly amortize the planning costs as well as improve reactive planning.

2.5.3 Reinforcement Learning Based

Reinforcement Learning is a machine learning algorithm that teaches a model to take appropriate actions by using the concepts of rewards and punishment. In general, reinforcement learning agents understand and interpret their environment, make decisions, and learn through trial and error. To maximize long-term benefits, determining the best sequence of actions is the ultimate objective. Reinforcement learning approaches can be classified as value-based, policy-based, or participant-criticized approaches. Actor-critic is derived from policy-based methods that use critics to estimate action value.

In [77], the authors proposed a Soft actor-critic based path planning algorithm for manipulators in environments that have periodically moving obstacles. Specifically, deep neural networks are created so that they can utilize position information of moving obstacles over a finite time period. Furthermore, the hindsight experience replay technique [78] is used to make efficient use of the training data.

In [79], they presented the actor-critic based algorithm that is based on deterministic policy gradients and can operate over continuous action spaces. Their experimental results have shown that their algorithm can solve challenging problems in various domains, including manipulation.

In [80], they presented a method in which multiple robots can learn a policy-based robotic strategy cooperatively with deep reinforcement learning. Specifically, using a deep reinforcement learning approach based on off-policy training of deep Q-functions, they demonstrated that a recent deep reinforcement learning algorithm can perform complex 3D manipulation tasks, and that deep neural network policies can be efficiently learned.

3.1 Approach of This Study

In most cases, the robot arm manipulators require rapid computing of efficient and smooth robot arm motions between the start and desired goal configurations in the dynamic environment. The deep learning method is one approach for high-dimensional motion planning problems in the complex dynamic environment. We are planning to employ the supervised learning method for this research. Most specifically, we are planning to develop a new deep artificial neural network (ANN) or deep neural network (DNN) structure [9] for motion planning problems of the manipulators in dynamic environments.

There are several types of neural networks in deep learning. such as convolutional neural networks (CNN), recurrent neural networks (RNN), fully connected neural networks (FCNN), etc. Those deep learning ANN structures have been changing the way we interact with the world. Because those ANN can solve many problems successfully with a high accuracy rate. These different types of neural network structures are at the core of the deep learning revolution. They work well in many applications like image recognition, classification, speech recognition, auto-control etc. For applications that contain images or videos, CNNs work pretty well [29,30,31,32]. RNNs are a kind of neural network that specializes in processing sequences. RNNs are often used in Natural Language Processing (NLP) tasks since they often deal with sequence data. However, in our project, moving the robot arm from the starting configuration to the goal configuration can be considered as the time series action. So, we can use the RNN as the neural network structure.

In general, it's an online execution of O (1) time and space complexity [5] for testing the ANN, which would solve the problem effectively. Due to the time and space complexity of the ANN structure, we decided to employ the RNN structure for this research.

3.1.1 Limitation of existing supervised learning methods

Compared with [51] and [52], we consider more in our neural network structure which can handle motion planning problems in dynamic environments with moving obstacles. [51] and [52] do not actually address the moving obstacles in their example. Our work is to close the gap of [51] and [52] for motion planning problems in dynamic environments. Moreover, [52] does not consider the stand-alone shape information during training and testing their ANN structure. The shape information of a robot is less important in simple environments. This may be the reason that led to the success of [52] in simple environments. But suppose we apply the same robot to complex environments. In that case, the shape information will become very important. If we do not consider the shape info, the ANN may fail to predict the output path in the complex environment. The training process of ANN with joint angles information(configurations) will not guarantee the ANN learns the actual shape info of the robot. However, training the ANN will DH parameters and moment of inertia can guarantee the ANN learns the shape info of the robot, which can improve the performance of the ANN.

So, the algorithm in [52] may fail in dynamic environments. The obstacles flowing around will lead the problem to be much more complicated because the dynamic environment contains a series of scenario one environments mentioned above.

Due to the limitations of [52] in the above paragraph, we consider two more points:

1. We consider the velocity of robot and obstacles in our ANN structures. So that it can output a path that can predict the movement of obstacles and avoid the moving obstacles.

We believe that, given the previous configuration and current configuration information as input, the ANN can enrich the knowledge of the ANN to generalize the learning ability to infer speed information. The time interval between two consequent sample configurations during data acquisition is assumed to be a constant Δt. The ANN structure can infer the speed of the robot and obstacles during the training process. As the result, the ANN structure can predict the position of the robot and obstacles at the future time t+∆t given necessary information of the current time t and previous time t-∆t as input. Consider the following example: the robot needs to go to the goal position as shown in the following picture. The blue color rectangle shape blocks are the static obstacles. The orange circle is the moving obstacle will move back and forth periodically. The arrows are the moving direction of the objects. The green color square is the robot. And it located at the initial position at time t. The light green color square is the goal. The robot needs to move to the goal position without colliding with all the obstacles.

If the ANN does not consider the previous environmental information, it may output a path as shown in the figure below, causing the robot to collide with a moving obstacle. This happened because the ANN does not understand the motion of the circular shape obstacle or the robot itself.

In contrast, if the ANN has the previous information at t- Δt about the environment and is trained, it can output the feasible path which will avoid the moving obstacle like the following picture:

2. We consider the stand-alone shape and mechanical structure information of the robot in our ANN structures. Robot shape information is important when planning in complex environment. We assume the DH parameters [68] and moment of inertia can represent the shape information of the robot.

The joint angles information can represent the shape info of the robot in some cases. But training the ANN only with joint angles info may not guarantee the ANN learns the entire shape information of the robot. Whereas taking the stand-alone shape info and fitting them in the ANN will remove the uncertainty of getting the shape info of robot. So, in our project, we will consider the shape or mechanical structure of the robot when building the ANN, and test its effect on the accuracy of the plan.

The following preliminary materials will be used in this research: including the Dienavit-Hartenberg (DH) parameters [68] and moment of inertia.

3.2 Denavit–Hartenberg Parameters

As a first step in analyzing the collision avoidance problem of the robotic arm or manipulator, its kinematics must be examined in order to establish the posture relationships between its joints and links.

In a robotic arm, the forward kinematics is determined by connecting a frame of reference to each link of an open chain and then calculating the forward kinematics from the relative displacement between adjacent links. The Dienavit-Hartenberg (DH) convention [68] allows for the derivation of variables and parameters from a kinematic chain of the robot manipulator. And the kinematic and dynamic model of a manipulator can be obtained after using these parameters and variables. These parameters and variables are called DH parameters [68]. The DH parameters are usually provided by the manufacturer of serial robots as a way of describing the robot's architecture.

There are four variables that are used to define the coordinate frame for every link in a robotic arm using the DH convention: link length, link twist, link offset, and joint angle (ai, αi, di, θi). Table 1 is the Kinova Gen3 7 DOF spherical classical DH parameters table [12]. The DH parameters will be used in the ANN as the constants in this study.

Table . The Kinova Gen3 7 DOF spherical classical DH parameters.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i | αi (radians) | ai (m) | di (m) | θi (radians) |
| 0 (from base) | π | 0 | 0 | 0 |
| 1 | π/2 | 0 | -0.2848 | q1 |
| 2 | π/2 | 0 | -0.0118 | q2 +π |
| 3 | π/2 | 0 | -0.4208 | q3 +π |
| 4 | π/2 | 0 | -0.0128 | q4 +π |
| 5 | π/2 | 0 | -0.3143 | q5 +π |
| 6 | π/2 | 0 | 0 | q6 +π |
| 7(to interface) | π | 0 | -0.1674 | q7 +π |

q1, q2... refers to the joint angles.

3.3 Moment of Inertia

The moment of inertia of an object, or moment of rotation, is a concept used to describe a rigid body that rotates about a fixed axis. The moment of inertia of an object mass located a distance from the center of rotation is defined by:

And the point mass of inertia is:

Because the moment of inertia of an ordinary object represents a continuous distribution of mass, it is usually calculated using calculus, which is capable of handling continuous variables. So, the moment of inertia of an ordinary object can be written in the integral format:

The values of the moment of inertia for the common object shapes around specified axes can be calculated easily by using the formulas.

Table 2 shows the six distinct moments of inertia (Ixx, Ixy, Ixz, Iyy, Iyz, Izz) of the Kinova Gen3 7 DOF robot with vision module [12]. The Table 2 variables will convert as the numpy array [69] and feed in the ANN in this research.

Table . Moments of inertia of the Kinova Gen3 7 DOF robot with vision module.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Physical quantity | moments of inertia (kg · m2) | | | | | |
| Ixx | Ixy | Ixz | Iyy | Iyz | Izz |
| Base | 0.004622 | 0.000009 | 0.00006 | 0.004495 | 0.000009 | 0.002079 |
| Link 1 | 0.00457 | 0.000001 | 0.000002 | 0.004831 | 0.000448 | 0.001409 |
| Link 2 | 0.011088 | 0.000005 | 0 | 0.001072 | -0.00069 | 0.011255 |
| Link 3 | 0.010932 | 0 | -0.000007 | 0.011127 | 0.000606 | 0.001043 |
| Link 4 | 0.008147 | -0.000001 | 0 | 0.000631 | -0.0005 | 0.008316 |
| Link 5 | 0.001596 | 0 | 0 | 0.001607 | 0.000256 | 0.000399 |
| Link 6 | 0.001641 | 0 | 0 | 0.00041 | -0.00028 | 0.001641 |
| Link 7 | 0.000587 | 0.000003 | 0.000003 | 0.000369 | 0.000118 | 0.000609 |

3.4 Simulated Environment

In this study, the Kinova Gen3 with the vision module robot will be used in the simulated environment. Experimental data will be generated in the simulated environments and then recorded in the files. The experimental data includes the environment number, robot movement trajectories (configurations), and obstacles information such as position and radius. We will generate 100 different environments for the robot with random initial and goal configurations and randomly sized obstacles. For each environment, we will generate 5000 motion paths. The setup details of each environment in Klampt [70] will be as follows:

Terrain Models: Terrain models are very simply collision geometries that include friction coefficients. The collision check will not apply to terrain models. The ground and the brown rectangle rod are the terrain models. The positions of these two terrain models are fixed in the environment.

Robot: One Kinova Gen3 with the vision module will be placed at the origin (0, 0, 0) of the Euclidean coordinate system. The robot also sits on top of the brown rectangle rod. The pose of the robot is randomly generated by the system for both its initial and goal configurations. The robot in Figure 20 is the initial configuration.

Obstacles: There are ten rigid spherical obstacles. Including two moving obstacles, and the rest of them are static obstacles. All the spherical obstacles will be inside the robot’s workspace. The position and radius of the sphere are randomly generated for each environment. To distinguish moving obstacles from static ones, we color the two kinds of obstacles differently. The green colored spheres are the moving obstacles, while the gray ones are the static obstacles. Figure 20 shows 2 green and 8 gray random size spheres. The position and radius information of the spheres of the example environment shown in Figure 20 can be viewed in Table 3.

Units: The units of length are meters, and the angles in the configuration are radians in the program.

Figure 20 is one sample environment for the experiment.

Figure . Figure 20. One Simulated Environment in Klampt. There are ten obstacles, two moving spheres in green and eight static spheres in gray. The Kinova Gen3 with vision module robot is sitting at (0, 0, 0) position with a random initial configuration. The brown color rectangle rod and the ground are the terrain models.

Table . Example obstacles information in environment#10

|  |  |  |
| --- | --- | --- |
| Obstacles information | Type | location and radius (first 3 parameters are the location information, x, y, z distance respectively, and the last parameter is the radius) |
| Obstacles (Spheres) | Moving | [-0.73, -0.67, -0.24, 0.19] |
| Moving | [0.35, -0.69, -0.43, 0.19] |
| Static | [0.93, -0.27, 0.08, 0.17] |
| Static | [0.13, -0.59, 0.35, 0.2] |
| Static | [0.3, -0.03, 0.08, 0.14] |
| Static | [-0.85, 0.65, -0.39, 0.25] |
| Static | [0.9, -0.38, 0.29, 0.21] |
| Static | [0.68, -0.81, 0.27, 0.17] |
| Static | [0.19, 0.7, -0.7, 0.21] |
| Static | [0.26, -0.38, -0.53, 0.19] |

3.5 Data Acquisition/Data Preparation

As a recap, we are planning to develop a deep learning based artificial neural network structure for this study. An artificial neural network will be developed to be used to extend motion planning from a static to a dynamic environment through its generalizing capability. And data collection is one of the most crucial aspects of machine learning and deep learning algorithms.

Data plays an important role in machine learning [71]. The training process would not be possible without the data. So, it is critical for training the ANN to be successful. Data preparation is the step of collecting and cleaning the data. So, data preparation is a critical and important step in machine learning and deep learning algorithms, as it affects the success of the ANN enormously.

The deep learning model will become more complex when you want to learn more parameters or are attempting add more complex lays to the DNN structure. You will also have to train the DNN model with more data. Because it is essential to have a large dataset for the DNN model to smooth out fluctuations and generate accurate interpretations [72]. So, we need to collect more data.

In supervised learning, the accuracy of DNN structural models is not only highly dependent on the large dataset but also highly dependent on the quality of the data. The quality of the data refers to the optimality of the sample data in this study. So, in this study, we need to collect a lot of valuable data that is captured in good quality, structured and cleansed.

Klampt [70] is an open-source, cross-platform software package for robot modeling, simulating, planning, optimization, and visualization. In this study, the Klampt package will be used to generate the training data for the DNN structure as well as the dynamic environment simulating for the articulated robot.

Klampt has built-in motion planning algorithms for the robots, including some of the most popular ones such as RRT, RRT\*, PRM, PRM\*, etc. As we discussed earlier in the related work section, RRT\* is the asymptotically optimal algorithm. especially in areas with dense obstacles, the RRT\* algorithm provides a good solution for static environments. So, the built-in RRT\* algorithm in Klampt will be employed to generate motion data in this study. Klampt also provides some industry and research robot models such as Baxter, PR2, PUMA760 robots. Finally, the Kinova Gen3 with vision module robot will be used in the simulated dynamic environment that created by Klampt. Table 4 shows one sample of complete trajectory of the robot that generate from the built-in RRT\* in Klampt, and there are 5 milestones (waypoints) in this path/ trajectory. The first waypoint (waypoint 1) is the initial configuration, and the last waypoint (waypoint 5) is the goal configuration. Figure 21 shows one example of the robot’s trajectory, from initial to goal configurations.

Table . Sample trajectory of environment # 25, initial and goal configurations were randomly generated. A Kinova gen3 robot model was used to generate trajectories.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Configurations  (In radians) | waypoint 1 | waypoint 2 | waypoint 3 | waypoint 4 | waypoint 5 |
| Joint 1 | 2.393870246 | 2.393870246 | 3.806530781 | 4.349683035 | 4.349683035 |
| Joint 2 | -0.791958832 | -0.791958832 | 1.233096018 | 2.011707071 | 2.011707071 |
| Joint 3 | 3.517702516 | 3.517702516 | 4.677142704 | 5.122934553 | 5.122934553 |
| Joint 4 | -2.212035026 | -2.212035026 | 1.136999242 | 2.424665648 | 2.424665648 |
| Joint 5 | -5.9771968 | 0.305988507 | 1.101498416 | 1.407363123 | 1.407363123 |
| Joint 6 | -0.575258456 | -0.575258456 | -0.35904457 | -0.275912738 | -0.275912738 |
| Joint 7 | 1.999459801 | 1.999459801 | 1.028849979 | 0.655661304 | -5.627524003 |

Figure . Render the environment in Klampt with the robot’s movement trajectory.

We are planning to generate the training data for both static and dynamic environments.

3.5.1 Static case

For the static case, we are planning to generate 100 different environments that each environment has one Kinova Gen3 robot in origin (0, 0, 0) and has ten rigid spherical obstacles. In each of the environments, the position of the obstacles will be generated by the program within the robot’s workspace. The radius of spherical obstacles is also generated by the program within a certain range. Here, we set the range as from 0.05 to 0.35 meters. The initial configuration and goal configuration of the robot are randomly generated by the program. These two parameters will be passed on to the motion planning algorithm as input. And then, the program will run the RRT\* algorithm as the motion planning algorithm for the pre-setup environment to generate the robot trajectory data, and the output data will be recorded in the text files. In each of the environments, we are planning to generate 5000 different sets of trajectories for the robot. However, we are not guaranteeing that they are all different because we are using the random function to generate the initial and goal configurations. It’s possible to have two or more sets of the same initial and goal configurations. In the data generating phase, as mentioned before, we will generate 5000 sets of trajectories. These 5000 sets of data will be generated in parallel. We will employ the parallel computing technique for running each of the RRT\* algorithms in one environment so that we can obtain the data quickly. Finally, the for loop will be used to go over all the 100 environments to generate the trajectories. All generated trajectories data will be recorded as the text files in the same output folder.

3.5.2 Dynamic case

Generating the training data for dynamic environments is a complicated task. Since there is no existing algorithm that can really handle the motion planning problems in dynamic environments for high DOF robotic arms. In this case, we are planning to generate training data from the simulated environments by using the following two strategies:

1. Generate the training data manually.

2. Generate the training data using the existing sampling-based algorithms.

Here, we assume the trajectories of the moving obstacles are known during the training stage. And then consider the entire trajectory of a single obstacle as one big obstacle in the static environment and use the existing sampling-based algorithms such as RRT\* to generate the training data.

3.6 Deep Learning Structure

We are planning to develop the RNN architecture for this project. This kind of neural networks is designed specifically to process sequences. Since they usually deal with sequence data, RNNs are often used in NLP. In this research, the trajectory of the robot motion is the time sequence data too. Therefore, we are planning to use RNN.

Also, we need an RNN model that can handle the Sequence-to-sequence (Seq2Seq) problem. Seq2Seq is the process of taking a sequence of items and transforming them into another sequence of items. Here, the length of the sequence of items is not a fixed number, and it can be changed for both input and output. The normal feed forward NN can only map one input (fixed length input) to one output. But the Seq2Seq model is a network structure that can map one to many, many to many, and many to one. The motion planning problems are obviously Seq2Seq problems. Ideally, these models would have both an encoder and a decoder. During the encoder stage, a hidden state vector captures the context of the input sequence and sends it to the decoder as a hidden state vector. The decoder produces the output sequence from those hidden state vectors.

The long short term memory network (LSTM) [73] is an extension for recurrent neural networks that basically add memory capacity. The LSTMs, which are the RNNs that can remember input over an extended period of time. Later, a novel architecture called Transformer [74] was proposed, which performed well in NLP. The main mechanism in Transformer is the attention module [74]. An attention-mechanism analyzes a sequence of inputs and determines which parts are important at each stage. The attention module can repeat its computations in parallel; this makes its computations fast.

LSTM and Transformer are both architectures that transform one sequence into another with two parts (encoder and decoder).

In this research, we plan to develop two DNN structures based on LSTM and Transformer, respectively. Then compare the performance of these two different models and pick the best one for our project. DH parameters and moment of inertia, which will be put together form an n-dimensional array as the constant input (partial) for the DNN. The following data will be concatenated together and provided as input to the DNN: current configuration (or initial configuration) of the robot, the previous of current configuration, target configuration (goal), current and previous obstacles information and the array of constants. The output is the predicted configuration with the given current configuration as input. The program will recursively predict the next configuration. The program will terminate when the last configuration reaches the goal configuration. The dataset will separate as 3 parts for the training process: training dataset, validation dataset, testing dataset. 80% of the data will be treated as training data; 10% will be validation data; and the rest of 10% will be testing data.

3.7 Evaluation

Evaluation is an important stage of the experiment. The following evaluation [11] metrics will be used for comparing the experiment result: success rate & planning time.

3.8 Other Hardware and Software Maybe Used

In this research, we are working on the 7-DOF (degree of freedom) Kinova Gen3 [12] robotic arm. And there are a lot of software that can be used in this research. We can also use Robot Operating System (ROS) gazebo [6] to simulate the robot arm and obstacles in the virtual environment. For the ANN structure and controlling the robot arm, we are going to use Python as the primary program language. For ANN structure, two popular deep learning frameworks will be used in this research: PyTorch [7] and TensorFlow [10].

3.9 Novelty of This Research:

1. We plan to develop new LSTMs, transformer-like RNN structures, for motion planning of articulated robots.

2. Add Moment of inertia as the constant input for the NN.

3. Add DH parameters as input for the NN.

4. We plan to implement the developed DNN structure for dynamic environments.

Reference

[1] LaValle, Steven M. Planning algorithms. Cambridge university press, 2006.

[2] Mohanan, M. G., and Ambuja Salgoankar. "A survey of robotic motion planning in dynamic environments." Robotics and Autonomous Systems 100 (2018): 171-185.

[3] Noreen, Iram, Amna Khan, and Zulfiqar Habib. "A comparison of RRT, RRT\* and RRT\*-smart path planning algorithms." International Journal of Computer Science and Network Security (IJCSNS) 16.10 (2016): 20.

[4] Lindemann, Stephen R., and Steven M. LaValle. "Current issues in sampling-based motion planning." Robotics research. The eleventh international symposium. Springer, Berlin, Heidelberg, 2005.

[5] Qureshi, Ahmed H., et al. "Motion planning networks." 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019.

[6] Chitta, Sachin, et al. "ros\_control: A generic and simple control framework for ROS." The Journal of Open Source Software 2.20 (2017): 456-456.

[7] Ketkar, Nikhil. "Introduction to pytorch." Deep learning with python. Apress, Berkeley, CA, 2017. 195-208.

[8] Ioan A. Sucan and Sachin Chitta, "MoveIt", [Online] Available at moveit.ros.org.

[9] Abiodun, Oludare Isaac, et al. "State-of-the-art in artificial neural network applications: A survey." Heliyon 4.11 (2018): e00938.

[10] Abadi, Mart&#x27;in, Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., … others. (2016). Tensorflow: A system for large-scale machine learning. In 12th $USENIX$ Symposium on Operating Systems Design and Implementation ($OSDI$ 16) (pp. 265–283).

[11] Gillani, Muhayyuddin, Aliakbar Akbari, and Jan Rosell. "Physics-based motion planning: Evaluation criteria and benchmarking." Robot 2015: Second Iberian Robotics Conference. Springer, Cham, 2016.

[12] "Technical Resources Library." Kinova. Web.

[13] Vemula, A., Muelling, K., & Oh, J. (2016, June). Path planning in dynamic environments with adaptive dimensionality. In Ninth Annual Symposium on Combinatorial Search.

[14] Stentz, Anthony. "Optimal and efficient path planning for partially known environments." Intelligent unmanned ground vehicles. Springer, Boston, MA, 1997. 203-220.

[15] Dechter, Rina, and Judea Pearl. "Generalized best-first search strategies and the optimality of A." Journal of the ACM (JACM) 32.3 (1985): 505-536.

[16] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001). "Section 24.3: Dijkstra's algorithm". Introduction to Algorithms (Second ed.). MIT Press and McGraw–Hill. pp. 595–601. ISBN 0-262-03293-7.

[17] LaValle, Steven M. "Rapidly-exploring random trees: A new tool for path planning." (1998): 98-11.

[18] Kuffner, James J., and Steven M. LaValle. "RRT-connect: An efficient approach to single-query path planning." Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065). Vol. 2. IEEE, 2000.

[19] Gammell, Jonathan D., Siddhartha S. Srinivasa, and Timothy D. Barfoot. "Informed RRT\*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic." 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2014.

[20] Karaman, Sertac, and Emilio Frazzoli. "Sampling-based algorithms for optimal motion planning." The international journal of robotics research 30.7 (2011): 846-894.

[21] Likhachev, Maxim, et al. "Anytime Dynamic A\*: An Anytime, Replanning Algorithm." ICAPS. Vol. 5. 2005.

[22] Aine, Sandip, and Maxim Likhachev. "Truncated incremental search." Artificial Intelligence 234 (2016): 49-77.

[23] Miao, Hui, and Yu-Chu Tian. "Robot path planning in dynamic environments using a simulated annealing based approach." *2008 10th International Conference on Control, Automation, Robotics and Vision*. IEEE, 2008.

[24] Aurenhammer, Franz, and Rolf Klein. "Voronoi Diagrams." *Handbook of computational geometry* 5.10 (2000): 201-290.

[25] Likhachev, Maxim, Geoffrey J. Gordon, and Sebastian Thrun. "ARA\*: Anytime A\* with provable bounds on sub-optimality." *Advances in neural information processing systems* 16 (2003): 767-774.

[26] Kavraki, Lydia E., et al. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces." *IEEE transactions on Robotics and Automation* 12.4 (1996): 566-580.

[27] Bohlin, Robert, and Lydia E. Kavraki. "Path planning using lazy PRM." *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 1. IEEE, 2000.

[28] Raajan, Jeevan, et al. "Real Time Path Planning of Robot using Deep Reinforcement Learning." *IFAC-PapersOnLine* 53.2 (2020): 15602-15607.

[29] Deng, Jia, et al. "Imagenet: A large-scale hierarchical image database." *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009.

[30] Girshick, Ross. "Fast r-cnn." *Proceedings of the IEEE international conference on computer vision*. 2015.

[31] He, Kaiming, et al. "Mask r-cnn." *Proceedings of the IEEE international conference on computer vision*. 2017.

[32] Theckedath, Dhananjay, and R. R. Sedamkar. "Detecting affect states using VGG16, ResNet50 and SE-ResNet50 networks." *SN Computer Science* 1.2 (2020): 1-7.

[33] Schwartz, Jacob T., and Micha Sharir. "On the “piano movers” problem. II. General techniques for computing topological properties of real algebraic manifolds." *Advances in applied Mathematics* 4.3 (1983): 298-351.

[34] Lozano-Perez, Tomas. "Automatic planning of manipulator transfer movements." *IEEE Transactions on Systems, Man, and Cybernetics* 11.10 (1981): 681-698.

[35] Lozano-Pérez, Tomás, and Michael A. Wesley. "An algorithm for planning collision-free paths among polyhedral obstacles." *Communications of the ACM* 22.10 (1979): 560-570.

[36] Planning, Spatial. "A Configuration Space Approach." *IEEE Trans, on Computers (C* 32 (1983): 108-120.

[37] Kamon, Ishay, Elon Rimon, and Ehud Rivlin. "Tangentbug: A range-sensor-based navigation algorithm." *The International Journal of Robotics Research* 17.9 (1998): 934-953.

[38] Lumelsky, Vladimir J., and Alexander A. Stepanov. "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape." *Algorithmica* 2.1 (1987): 403-430.

[40] Elbanhawi, Mohamed, and Milan Simic. "Sampling-based robot motion planning: A review." *Ieee access* 2 (2014): 56-77.

[41] Sciavicco, Lorenzo, and Bruno Siciliano. "A solution algorithm to the inverse kinematic problem for redundant manipulators." *IEEE Journal on Robotics and Automation* 4.4 (1988): 403-410.

[42] Salzman, Oren, and Dan Halperin. "Asymptotically-optimal motion planning using lower bounds on cost." *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015.

[43] Otte, Michael, and Emilio Frazzoli. "$${\mathrm {RRT^{X}}} $$: Real-Time Motion Planning/Replanning for Environments with Unpredictable Obstacles." *Algorithmic Foundations of Robotics XI*. Springer, Cham, 2015. 461-478.

[44] Khatib, Oussama. "Real-time obstacle avoidance for manipulators and mobile robots." *Autonomous robot vehicles*. Springer, New York, NY, 1986. 396-404.

[45] Volpe, Richard, and Pradeep Khosla. "Manipulator control with superquadric artificial potential functions: Theory and experiments." *IEEE Transactions on Systems, Man, and Cybernetics* 20.6 (1990): 1423-1436.

[46] Kim, Jin-Oh, and Pradeep Khosla. "Real-time obstacle avoidance using harmonic potential functions." (1992).

[47] Hauser, Kris. "Lazy collision checking in asymptotically-optimal motion planning." *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015.

[48] Naderi, Kourosh, Joose Rajamäki, and Perttu Hämäläinen. "RT-RRT\* a real-time path planning algorithm based on RRT." *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*. 2015.

[49] Yuan, Chengren, et al. "A heuristic rapidly-exploring random trees method for manipulator motion planning." *IEEE Access* 8 (2019): 900-910.

[50] Lorenz, Edward N. "Deterministic nonperiodic flow." *Journal of atmospheric sciences* 20.2 (1963): 130-141.

[51] Qureshi, Ahmed Hussain, et al. "Motion planning networks: Bridging the gap between learning-based and classical motion planners." *IEEE Transactions on Robotics* 37.1 (2020): 48-66.

[52] Bency, Mayur J., Ahmed H. Qureshi, and Michael C. Yip. "Neural path planning: Fixed time, near-optimal path generation via oracle imitation." *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019.

[53] Gai, Sheng Nan, et al. "6-DOF robotic obstacle avoidance path planning based on artificial potential field method." *2019 16th International Conference on Ubiquitous Robots (UR)*. IEEE, 2019.

[54] Lin, Hsien-I., and Ming-Feng Hsieh. "Robotic arm path planning based on three-dimensional artificial potential field." *2018 18th International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2018.

[55] Kim, Jin-Oh, and Pradeep Khosla. "Real-time obstacle avoidance using harmonic potential functions." (1992).

[56] Abhishek, Talabattula Sai, Daniel Schilberg, and Arockia Selvakumar Arockia Doss. "Obstacle Avoidance Algorithms: A Review." *IOP Conference Series: Materials Science and Engineering*. Vol. 1012. No. 1. IOP Publishing, 2021.

[57] Palmieri, Giacomo, and Cecilia Scoccia. "Motion planning and control of redundant manipulators for dynamical obstacle avoidance." *Machines* 9.6 (2021): 121.

[58] Zhang, Hui, et al. "Analysis of Obstacle Avoidance Strategy for Dual-Arm Robot Based on Speed Field with Improved Artificial Potential Field Algorithm." *Electronics* 10.15 (2021): 1850.

[59] Ichter, Brian, and Marco Pavone. "Robot motion planning in learned latent spaces." *IEEE Robotics and Automation Letters* 4.3 (2019): 2407-2414.

[60] Kurutach, Thanard, et al. "Learning plannable representations with causal infogan." *Advances in Neural Information Processing Systems* 31 (2018).

[61] Pfeiffer, Mark, et al. "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots." *2017 ieee international conference on robotics and automation (icra)*. IEEE, 2017.

[62] Hamandi, Mahmoud, Mike D’Arcy, and Pooyan Fazli. "Deepmotion: Learning to navigate like humans." *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2019.

[63] Reif, John H. "Complexity of the mover's problem and generalizations." *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*. IEEE Computer Society, 1979.

[64] Arslan, Oktay, and Panagiotis Tsiotras. "Use of relaxation methods in sampling-based algorithms for optimal motion planning." *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013.

[65] Otte, Michael W. "A survey of machine learning approaches to robotic path-planning." *University of Colorado at Boulder, Boulder* (2015).

[66] Qureshi, Ahmed Hussain, et al. "Constrained motion planning networks x." *IEEE Transactions on Robotics* (2021).

[67] Yang, Ge, et al. "Plan2vec: Unsupervised representation learning by latent plans." *Learning for Dynamics and Control*. PMLR, 2020.

[68] Denavit, Jacques, and Richard S. Hartenberg. "A kinematic notation for lower-pair mechanisms based on matrices." (1955): 215-221.

[69] Van Der Walt, Stefan, S. Chris Colbert, and Gael Varoquaux. "The NumPy array: a structure for efficient numerical computation." *Computing in science & engineering* 13.2 (2011): 22-30.

[70] Hauser, Kris. Intelligent Motion Laboratory at Duke University, http://motion.pratt.duke.edu/klampt/.

[71] Kotsiantis, Sotiris B., Dimitris Kanellopoulos, and Panagiotis E. Pintelas. "Data preprocessing for supervised leaning." *International journal of computer science* 1.2 (2006): 111-117.

[72] Chen, X. W., and X. Lin. "IEEE Access 2, 514 (2014)." (2014).

[73] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

[74] Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

[75] Bylander, Tom. "The computational complexity of propositional STRIPS planning." Artificial Intelligence 69.1-2 (1994): 165-204.

[76] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems 27 (2014).

[77] Prianto, Evan, et al. "Deep reinforcement learning-based path planning for multi-arm manipulators with periodically moving obstacles." Applied Sciences 11.6 (2021): 2587.

[78] Kim, MyeongSeop, et al. "Motion planning of robot manipulators for a smoother path using a twin delayed deep deterministic policy gradient with hindsight experience replay." Applied Sciences 10.2 (2020): 575.

[79] Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." arXiv preprint arXiv:1509.02971 (2015).

[80] Gu, Shixiang, et al. "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates." 2017 IEEE international conference on robotics and automation (ICRA). IEEE, 2017.

Appendix

Wolfram Mathematica:

<https://www.wolframcloud.com/objects/demonstrations/RapidlyExploringRandomTreeRRTAndRRT-source.nb>

<https://demonstrations.wolfram.com/BreadthFirstSearchRobotMotionPlanning/>

Matlab for PRM:

<https://www.mathworks.com/help/robotics/ug/probabilistic-roadmaps-prm.html>

RoboDK:

<https://robodk.com/download>

Klampt:

<http://motion.cs.illinois.edu/software/klampt/latest/pyklampt_docs/>