Autonomous Motion Planning Hand

Gabriel Agostine GABRIEL.AGOSTINE@COLORADO.EDU
Department of Aerospace Engineering Sciences
University of Colorado Boulder

Abstract

The goal of this project is to allow a simulated hand and armature to autonomously guide itself from an initial starting configuration to a goal configuration. The system was designed to be modular, meaning that it is dynamic in the allowed number of fingers per hand and can have more than 5 fingers on the hand if desired. The hand and armature system must avoid collisions with other fingers and environmental obstacles, if specified, and efficiently explore its allowable configuration space to reach said goal state. This is done using a Rapidly Exploring Random Tree (RRT) Algorithm to explore the allowed composed configuration space for the hand and armature link system.

I. Introduction

With the rise of autonomous motion planning algorithms, many unsolvable problems have become solvable. Similarly, problems that were thought to be very complex in how they were coded can now be dynamically coded. A great example of this is with some form of a linked manipulator such as a human arm, which can be represented as a link manipulator with attached link manipulators. The need for autonomous human arms are widespread, be it Biomimetics, obstacle movement or various grappling tasks.

As the number of joints and allowed Degrees of Freedom (DoF) of the system increases, more edge cases would need to be defined to fully describe the system. For an n-link manipulator arm, the configuration space needed to describe the system in $\mathbb{R}^3$ would be $3n$. For the human hand, each finger has 4 degrees of freedom, yielding a 20 DoF configuration space. Adding in an armature with an elbow and shoulder joint each with 1 DoF, we have a full composed configuration space in $\mathbb{R}^{22}$.

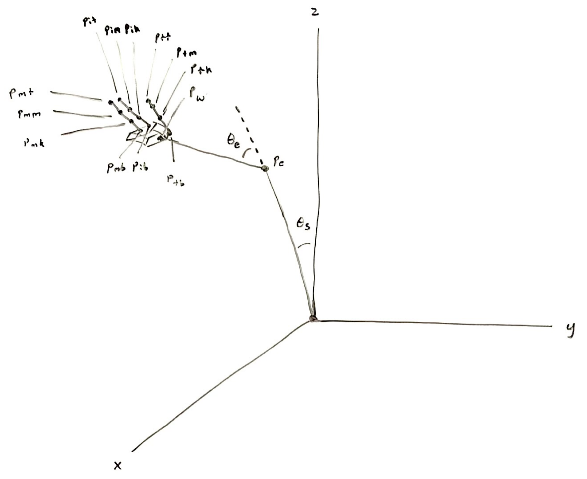

Fig. 1 Global Coordinate Frame

For a composed space this large, the best approach is to use some type of Sampling Based Motion Planner (SBMP). This allows us to more efficiently search the allowable configuration space and find a solution path. For our system, our configuration space is entirely angular (or radial) so we must also consider the radial or Non-Euclidean distance between configurations.

As described previously, the system must be modular, meaning it should be able to allow for any number of attached links given the geometry of the system. System properties such as hand and armature dimensions, allowed ranges for each configuration subspace and any other dimensional property must be easily changeable. The hand and armature must also avoid collisions with environmental obstacles and other parts of the system (such as finger-finger collision). These details will be further fleshed out in the Methodology section.

II. Problem Description

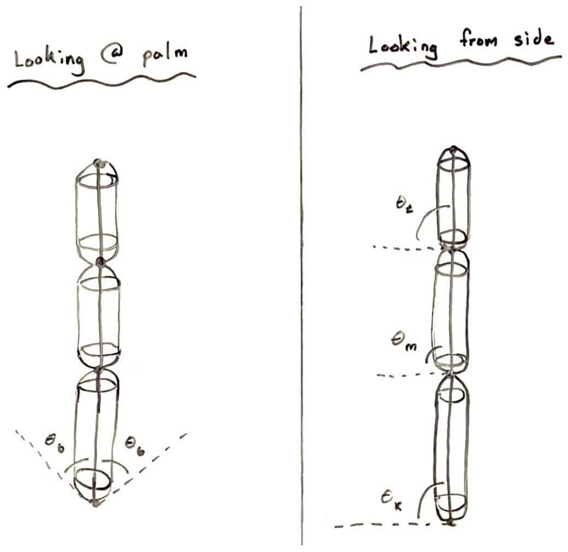

Fig. 2 Local Finger Angles

For a full simulated hand and armature system with full range of motion for all joints but the wrist, elbow and shoulder, we may use a RRT Algorithm to explore this composed configuration space. Each finger joint will be treated as its own agent in a Centralized RRT planning algorithm, where each joint is moved in sync with all other joints. We assume 4 DoF for each finger (5 fingers), no wrist movement, and 1 DoF for the elbow and shoulder around the Global y-axis. This yields a total composed configuration space of $\mathbb{R}^{22}$, which is non-visualizable.

The system may however be visualized by translating each configuration subspace value to a translation and rotation in the workspace ($\mathbb{R}^3$) using forward kinematics. The location of each joint can be backed out with this method, and plotted in the Global coordinate frame as seen in **Figure 1**. Relative to the shoulder of a human body, the x-axis will point from the right shoulder to the left shoulder along the clavicle, the y-axis will point forwards from the shoulder joint and the z-axis follows typical right-hand rule, pointing downwards from the shoulder along the leg. It should be noted that this system simulates the right hand and not the left, however due to the modular design of this system the left hand may be easily simulated.

Each finger, as mentioned, has 4 considered joints and 4 DoF. These are as follows:

Joint	Rotation Angle about Joint
Shoulder	θ_s
Elbow	θ_e
Wrist	$--$
Base	θ_b, θ_k
Knuckle	θ_m
Middle	θ_t
Tip	$--$

Table 1 Joint angles

where:

Angle	θ_{min} (Thumb) [rad]	θ_{max} (Thumb) [rad]	θ_{min} (else) [rad]	θ_{max} (else) [rad]
θ_s	$-\frac{\pi}{2}$	$\frac{\pi}{2}$	$-\frac{\pi}{2}$	$\frac{\pi}{2}$
θ_e	0	$\frac{3\pi}{4}$	0	$\frac{3\pi}{4}$
θ_b	0	$\frac{\pi}{4}$	$-\frac{\pi}{9}$	$\frac{\pi}{9}$
θ_k	$-\frac{\pi}{4}$	$-\frac{\pi}{8}$	0	$\frac{\pi}{2}$
θ_m	0	$\frac{\pi}{4}$	0	$\frac{3\pi}{4}$
θ_t	0	$\frac{\pi}{2}$	0	$\frac{\pi}{2}$

Table 2 Allowable angle ranges for each joint

The angles defined in **Tables 1 and 2** are physically oriented as seen in **Figure 2**. They each rotate about a set axis in the Global (or local for transformation matrices) frame depending on the finger. The axes each θ rotates about are defined as such:

Angle	Thumb axes	Else axes
θ_s	X	X
θ_e	X	X
θ_b	Y	X
θ_k	X	Y
θ_m	X	Y
θ_t	X	Y

Table 3 Axes of rotation

Along with the definition of the allowable range of each subspace in our composed configuration and the angles necessary to fully describe the hand and armatures orientation in 3D space, we must also define the lengths of each joint in the hand. This is needed in order to account for the translation in the Homogeneous Transformation matrices used to calculate the position of each joint. These values are as follows:

Joint	Base [cm]	Middle [cm]	Tip [cm]
Arm	32	29	$--$
Palm	2.5	6	11
Thumb	7	4	3.5
Index	5	3	2.5
Middle	5.5	3.5	2.5
Ring	4.5	3.5	2.5
Little	4	2.5	2.25

Table 4 Joint Lengths for Hand and Arm

It should be noted that in **Table 4** the "Base" and "Middle" lengths, with respect to the arm, refer to the length of the Humerus and Forearm respectively. Similarly the lengths of all three dimensions, with respect to the palm, refer to the depth, width and height of the palm respectively (local x, y and z axes)[*]. These values allow us to now define how we account for the position of each joint given the values of our composed configuration space in $\mathbb{R}^{22}$.

III. Methodology

To determine the position of each joint, we use a series of Homogeneous Transformation Matrices that account for both rotation and translation relative to preceding joints. These matrices can be (in a general case) structured as follows[†]:

$$\begin{bmatrix} \underline{p}_{i+1} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{R}\left(\theta_\alpha, \theta_\beta, \theta_\gamma\right) & \underline{t} \\ \underline{0} & 1 \end{bmatrix} \begin{bmatrix} \underline{p}_i \\ 1 \end{bmatrix}$$

Here, $R\left(\theta_\alpha, \theta_\beta, \theta_\gamma\right)$ represents a rotation matrix constructed using a sequence of rotations about specific axes of the local frame. The rotation sequence can follow any convention, such as XYZ, ZYX, or even repeated axes like XYX, depending on the desired rotation order. This project assumed the XYZ order of rotation matrices, however any order will work with minor changes to needed θ values and their correspondent axes of rotation. The vector $\underline{t}$ denotes the corresponding translation of the local frame.

Lastly, $\underline{p}$ denotes the position of a point in space before and/or after the translation and rotation. It is important to note that when applying such Homogeneous Transformation matrices, the rotation of the local frame centered at $\underline{p}_i$ can be thought to occur first, then the translation from said local frame. This results in a new local frame centered at $\underline{p}_{i+1}$.

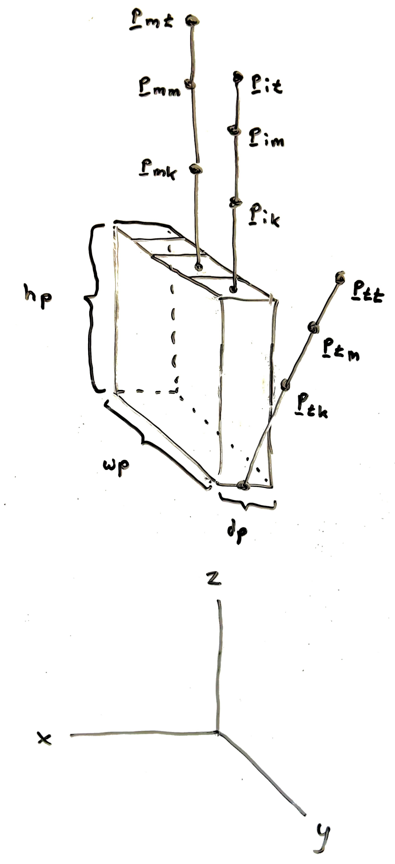

Fig. 3 Local Finger Orientation

By stringing together an arbitrary number of such transformation matrices, one may represent the location of a point in 3D space regardless of the number of joints (or links) in the system. In general, you will need $n + 1$ Transformation matrices for n joints from the Global origin to the desired point. For example, from the Global origin to the tip of any finger, 7 Transformation matrices were needed. These matrices followed the order of:

1) Don't rotate tip joint but translate local frame to middle joint
2) Rotate middle joint and translate local frame to knuckle joint

[*]All values were measured from the authors body
[†]Assuming the workspace is in $\mathbb{R}^3$

3) Rotate knuckle joint and translate local frame to base joint
4) Rotate base joint and translate local frame to wrist joint
5) Don't rotate wrist joint but translate local frame to elbow joint
6) Rotate elbow joint and translate local frame to shoulder joint (which is the Global frame)
7) Rotate shoulder joint but don't translate Global frame

The last step is to then multiply the location of the origin in Homogeneous coordinates $\left([\underline{0}, 1]^T\right)$ by the Transformation matrices defined above from $T_{1\rightarrow 7}$. This process can be modified slightly by reduction of matrices to calculate the position of any previous joint such as the base or wrist. Doing so allows us to find the location of any desired joint (as seen in **Figure 3**) in our 3D workspace.

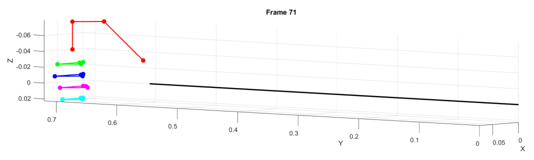

Fig. 4 Fist

It may be noted that Inverse Kinematics were not needed for this algorithm, as after specifying the desired starting and ending angles for each joint, we may back out the desired starting and ending positions using Forward Kinematics. This is also used to check for obstacles collision in the workspace and for plotting. Solving for the Inverse Kinematic equations would also prove to be a rather challenging task as well.

All that remains is the definition of our Planning Algorithm and how to hand obstacle collision. The former is a rather deceptively easy task. Although numerous Open-Source Motion Planning Libraries exist such as Open Motion Planning Library (OMPL), this project was coded entirely without the use of any external library or toolbox. It is coded in C++ and uses MATLAB to visualize the movement of the hand through a 3D workspace[‡].

This decision was made out of the desire to have no dependence on other toolboxes or external libraries for ease of use and for those who do not have access to such tools. The source code is provided at the end of this paper. Such decisions aside, the chosen algorithm was RRT, but design choices were made in order to accommodate a change in algorithm to a more optimal or efficient planner in the future.

In regards to other motion planners, Gradient Descent would work only in the presence of few obstacles, and with multiple fingers there exists numerous possibilities of accidentally encountering a local minima. Grid based planners such as wavefront simply cannot handle such a high dimensionality composed configuration space and is computationally infeasible. Probabilistic Roadmap planners may work, but are not as common nor have the heritage that Rapidly Exploring Random Tree (RRT) planners. Thus, the algorithm used was RRT, implemented following a standard RRT pseudocode:

```
Algorithm: RRT
Inputs: Environment, robot dimensions and starting parameters
Output: Path to goal
```

[‡]Animated videos and general results are covered in the Results section

```
1)  $T \leftarrow$ create tree structure and unordered map
2)  while (iteration < maxIteration) do
3)      $q_{rand} \leftarrow$ generate random sample (can be $q_{goal}$ for certain probability $\rho$)
4)      $q_{near} \leftarrow$ find closest node on $T$ to $q_{rand}$
5)      $q_{new} \leftarrow$ generate new point along line from $q_{near}$ towards $q_{rand}$
6)          if (isValid($q_{near}$, $q_{new}$)) then
7)              add $q_{new}$ to $T$
8)          if (inGoalRegion($q_{new}$)) then
9)              break
10) pathDijkstra($T$, $T.node[0]$, $T.node[T.node.size()]$) $\leftarrow$ returns path from root node to goal node
11) return path
```

where:

```
1) isValid($a$, $b$)
2)     for $\forall i \in [0 : numPoints]$
3)         $t = i/(numSteps - 1)$
4)         $c = (1 - t)\, a + (t)\, b$
5)         if (inObstacle($c$) || !inSS($c$))
6)             return false
7)     return true
```

Finger-finger collision detection was the main priority for this project, specifically focusing on allowing the hand to adapt to a large number of valid configurations and find a solution accurately and in a reasonable manner. Finger-finger collision was handled in a rather intuitive way, by considering the locations of joints on each finger.

By observing the human hand, we may find that we can define a cylinder around the length of each joint (with exception to the region of rotation) as seen in **Figure 2**. This makes noticeable the convenient property that if any two fingers knuckles are closer than the width of any finger then those fingers are in collision. The only exception to this rule is the thumb, where its motion allows it to more easily collide with other fingers, as well as its knuckle joint being disproportionately placed relative to the remaining fingers. However, the thumb may be accounted for by checking if each of its joints is a similar distance away from any of the other fingers joints, and if not then a collision exists.

Any further edges cases are handled by the inherent geometry of the hand, where the range of allowable joint angles prevents the finger from intersection with other fingers at all. This is much the case for finger-palm collision and humerus-forearm collision.

IV. Results

The results of this planner and its correspondent system are somewhat expected. Using a goal biasing probability of $p_{goal} = 1$, a iteration count of $it_{max} = 20000$, a step size of $r = 0.1$ and a goal acceptance region of $\epsilon = 0.01$ cm we achieve the following results when benchmarking 20 times with 5 fingers and an attached armature:

Parameter	Mean
Success rate [%]	100
Runtime [s]	13.7958

Table 5 Optimal 5 fingers and armature results

Parameter	Mean
Success rate [%]	80
Runtime [s]	975.9264

Table 6 5 fingers and armature results

The reason for choosing these specific simulation parameters is for testing for the time required for a simulation under optimal movement to find a solution, giving us an effective lower bound on computation time. A simulation with 5 fingers and a full armature takes too long to reasonably benchmark, as computation times can take upwards of 10-15 minutes each, so only 5 benchmarks were used.

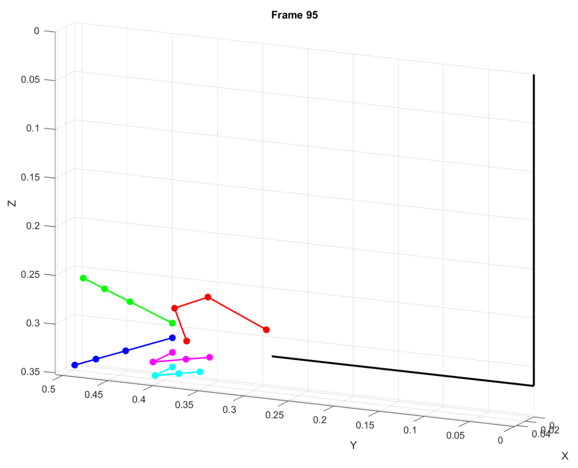

Fig. 5 Peace Sign

For a Centralized RRT planning algorithm with a composed configuration space in $\mathbb{R}^{22}$, these results are somewhat expected. Exploring a space of such high dimensionality is rather time consuming and (depending on the allowed step size and number of iterations) can result in some planning instances not succeeding.

This planner is naturally far from real time, which is the intuitive end goal of any planning algorithm. Since it is also a Sampling Based Motion Planner (SBMP) it inherits the properties of any ordinary RRT algorithm. It is thus only probabilistically complete, meaning as the number of allowed random samples (or it_{max}) goes to infinity, the probability of it finding a solution goes to 1. This is of course computationally infeasible. Similarly, it is not optimal, due to the nature of the standard RRT algorithm. However, it may be modified to become asymptotically optimal planner by means of extension into RRT* or RRT#. The latter option requires the use of a High-Level planner, which can be integrated into even the base RRT algorithm to help the planner more effectively search the composed configuration space and decrease runtime.

Using an external MATLAB code, we were able to visualize the motion of this planner when a successful solution was generated. Links to these videos are provided in the Appendix! Two different poses were commanded for the hand and armature to reach, the first being a peace sign and the second being an extended fist.

Looking back on the goals of this project before its start, we may consider if each of our design goals were achieved:

Design Goal	Achieved
1 finger	X
3 fingers	X
3 fingers, attached armature	X

Table 7 Design Goals

We achieved all the design goals of this project and more with a successful algorithm and system integration. Yet, there is always room for improvements, below is a list of desired future design goals external to the initial design goals:

Future Design Goal	Achieved
5 fingers	X
5 fingers, attached armature	X
5 fingers, attached armature, High-Level planner	– –
5 fingers, attached armature, High-Level planner, full DoF	– –
Real time response with the previous	– –
Physical/hardware system integration with the previous	– –

Table 8 Future Design Goals

V. Conclusion

This paper sought to analyze a Centralized RRT planner for the use in the autonomous motion control of a hand and armature for various Biomimetic, industrial and grappling tasks. We found the results of such to be far from real time, but promising in success rate. The system is modular, allowing for the removal and addition of fingers to the overall system. It is tolerant to a change in algorithm and, while not a completely optimized algorithm, helped the author better understand the C++ language and how to autonomously plan for high dimensional and very geometrically complex systems.

While this project was not coded using any External Libraries or Toolboxes, it still does run and will generate results, however due to the extreme complexity of the system and lack of efficient optimization of the proprietary code, it is hard to properly benchmark the system. Future steps will involve code optimization to isolate subroutines of longest runtime and optimization of them first. This paper regardless shows that this problem is not impossible and outlines the proper method on how to complete the task. More was achieved than was initially desired

VI. Acknowledgments

Special thank you to all involved with this project, technical or otherwise:

Morteza Lahijanian
Yusif Razzaq

VII. Appendix

5 finger with attached arm animation: Click Here

5 finger with attached arm animation: Click Here
Source code: Click Here