Robot Path Planning Using Newton-Raphson method Based on RRT-GD

Albert Author[1] and Bernard D. Researcher[2]

***Abstract*— We proposed a method on 7-arm redundant manipulator for obstacle-avoiding path planning. As we all know, RRT (rapidly-exploring random tree) method can perform well when doing obstacle-avoiding works. But the traditional RRT method concentrates mainly on the status space(S-space), it cant predict the end actuators trajectory, thus ending up with the separation between end actuator and robot joints. In this paper, we use RRT method in the configuration space(C-space), concentrating on both the end actuators status and the joints movements. In order to reduce the time for path planning, i.e., reduce the steps when doing RRT, we change RRT algorithm to be goal-direction (we call it RRT-GD), adapt to our common works. With this improve, we can see in this paper that result could be reached in less than 10 steps usually, while the original RRT algorithm needs often more than 100 steps doing the same task. In addition, we apply Newton-Raphson method to RRT to do inverse kinematics optimization, thus we can focus on both the S-space and C-Space. In the end, we use quintic polynomial to smooth the planning path.**

I. Introduction

As we all know, RRT (Rapidly-exploring random tree) is rapidly used in obstacle-avoiding path planning for redundant manipulator. It constructs a graph of obstacle free points in feasible space based on random sampling, in which we search the feasible path from the initial point to goal point. As a result of this, there are some great properties with RRT method compared to other obstacle-avoiding method:

- RRT method attends to explore unknown space.
- The graph constructed by RRT will gradually fill up the feasible space if the exploring times is enough.
- RRT method dont need precise modeling, instead, it only needs the information that if a point is obstacle free through collision detect. Thus ending up with the high efficiency of obstacle free planning.

Though it has these advantages, RRT will still perform low planning success rate when there is a great deal of obstacle surround, or the free degree of manipulator is high. In this way, Kuffner proposed a bidirectional RRT algorithm, bi-RRT, also called RRT-connect. Within this method, two trees were built based on the initial point and goal point separately. When expanding each time, the tree of initial point will extend to the other tree, or the contrary way, until the two trees meets. For it has goal heuristic, it improves the success rate of path planning.

*This work was not supported by any organization

[1]Albert Author is with Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, 7500 AE Enschede, The Netherlands albert.author@papercept.net

[2]Bernard D. Researcheris with the Department of Electrical Engineering, Wright State University, Dayton, OH 45435, USA b.d.researcher@ieee.org

In spite of these advantages of RRT and its modified algorithm, we still find it defective sometimes in doing our common works. As mentioned before, RRT attends to explore unknown space, so that it could exploring the whole free space. But we just dont need to explore the whole space when doing simply goal-reaching works. In this way, all we need to do is to find a feasible way from initial point to goal point without obstacle collision, i.e., we only need to explore a part of the whole space include initial point, goal point, and some obstacle. Thus we proposed RRT-GD (RRT with goal-directionality), using the train of thought above. As we can see later in this paper, by doing this way, we can get the result rapidly, usually ten times faster than the normal RRT method.

As we doing RRT in C-space (configuration space), we need to compute the joint angles of each point in the path tree, i.e. inverse kinematics optimization. There are also a lot of methods to do inverse kinematics optimization, e.g., GP (gradient projection method), WLN (weighted least norm method), extended Jacobi method, etc. In this paper, we adopt Newton-Raphson method to do this inverse kinematics optimization work. Compared to other methods, Newton-Raphson method performs faster and more accurate. Provided the result can get by Newton-Raphson method, it would take only less than 10 iteration times (usually 5 or 6 times), with bias less than micro-meters.

After we get the path from initial point to goal point, with detailed status information (pose & joint angles), we then use quintic polynomial to smooth the path [9]. We show our results in the experiment result part of this paper, and then make comparison with the other methods mentioned above.

We organize this paper according to our experiment process, to show our results more fluently. First, we discuss the basic mathematics knowledge and algorithm we use, include RRT and its modified method in more detail. Then, we introduce Newton-Raphson method in inverse kinematics by using the normal RRT method. After that, we present quintic polynomial method in smoothing path. Later, in order to accelerate the planning time and doing works real-time, we propose RRT-GD method, which press more close to our work in common use.

II. Experiment Environment

All the results we show is based on a redundant robot arm. Before introduce our algorithm, we can have a look at the basic environment.

A. Robot Model & Its Relevant Variables (Heading 2)

We use a 7-arm redundant robot manipulator to test the results in our experiment. We use the DH method (Denavit and Har- tenberg method) to model our robot arm, with the model axis built obeying D-H parameter method as the Fig.1 show.

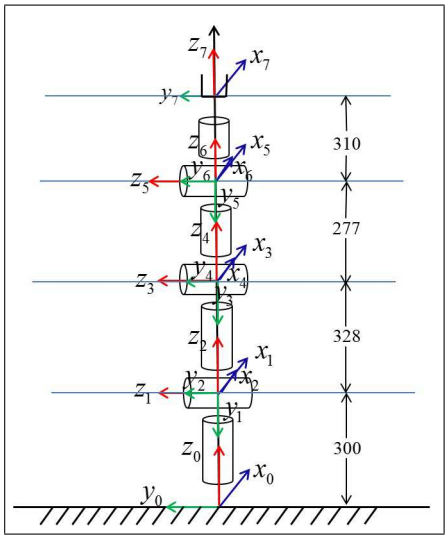

Fig. 1. Robot axis coordinates, obeying D-H parameter method, numbers in the graph is in milimeter.

As a result of this axis coordinates, we get the DH parameters as follow (show in table 1) :

TABLE I
DH PARAMETERS OF ROBOT.

Joint Number	θ_i	d_i(mm)	a_i(mm)	$\alpha_i(^\circ)$	area of $\theta_i(^\circ)$
1	q_1	300	0	-90	-180∼180
2	q_2	0	0	90	-90∼90
3	q_3	328	0	-90	-180∼180
4	q_4	0	0	90	-120∼120
5	q_5	277	0	-90	-180∼180
6	q_6	0	0	90	-120∼120
7	q_7	310	0	0	-180∼180

In table 1, θ means joint angle, the letter 'd' means connecting rod skew, 'a' means length of connecting rod, and α means tortuosity angle of connecting rod.

All our later works are based on table 1 and fig.1, including experiments and simulation. As we can see in fig.1, the joint angles from q_1 to q_7 are the free variables and the others are fixed within our experiments. So when we write rotation and translation matrix in homogeneous way, which is a 4×4 matrix, we can get the transformation equation from current coordinate (x_i, y_i, z_i) to the next coordinate (x_{i+1}, y_{i+1}, z_{i+1}) as below:

$$^{i-1}T_i = A_i = \begin{array}{l} Rot(z,\theta_i) \cdot Trans(0,0,d_i) \cdot \\ Trans(a_i,0,0) \cdot Rot(x,\alpha_i). \end{array} \quad (1)$$

Among (1), Rot(z, θ) means homogeneous transformation matrix when rotate the current coordinate θ degree around z-axis, and Trans(x, y, z) means homogeneous transformation matrix when transform the current cordiante with the vector $\vec{v} = (x, y, z)$.

Then, we can get the transformation matrix from base coordinate (x_0, y_0, z_0) to the end manipulator coordinate (x_7, y_7, z_7) as follow:

$$M = {}^0T_1 \, {}^1T_2 \ldots {}^6T_7 \quad (2)$$

B. Status Space & Configuration Space

Status Space is expressed with 7 parameters from q_1 to q_7, with the expression like this:

$$S = (q_1, q_2, \ \ldots \ , q_7) \quad (3)$$

As for the configuration space, i.e., pose space, we use Euler angles that obeys Euler Z-X-Z transformation rules to express the azimuth angles, which displays like $\gamma = (\psi, \theta, \phi)$. Position writes like $P = (x, y, z)$. Get the position and Euler angles together, we define the expression of pose like this:

$$X = (P, \gamma) = (x, y, z, \psi, \theta, \phi) \quad (4)$$

See that S is expressed with 7 free variables while X with 6, which tells the redundancy of our system, on which our work is based.

C. Forward Kinematics & Jacobian Matrix

As we can see in (2) & (3), M is decided by the current state S, thus the current pose X is also decided. When we transform X into homogeneous matrix defined as $H(X)$ and write M as $M(S)$ (means M is decided by S), we get the connection between S and X:

$$H(X) = M(S) \quad (5)$$

That is what our forward kinematics based on. Knowing the current S, we can get $M(S)$ from (2), then use (5) to get X. The detail operation from H(X) to X will not be discussed here, which can be easily found in primary textbook of robot kinematics. Combineg the process of turning $H(X)$ to X in M, we can finally get forward kinematics as below:

$$X = M(S) \quad (6)$$

The letter M in (6) is not the same meaning with (2) & (5), which combining the process from $H(X)$ to X. We can see M here in (6) as just a function with independent variables S and dependent variables X.

D. Inverse Kinematics & the Extend Inverse Jocobian Matrix

Inverse kinematics comes with the question: How could we get S when we know X? Since the system is reduntant, there may not be only one answer for S. There are methods from X to S directly using fixed angle, but we do inverse kinematics with the help of Jacobian matrix instead.

Looking back at (6), we write M in partial form:

$$X = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \end{bmatrix} = \begin{bmatrix} M_1(q_1, q_2, \ldots, q_7) \\ M_2(q_1, q_2, \ldots, q_7) \\ M_3(q_1, q_2, \ldots, q_7) \\ M_4(q_1, q_2, \ldots, q_7) \\ M_5(q_1, q_2, \ldots, q_7) \\ M_6(q_1, q_2, \ldots, q_7) \end{bmatrix} \tag{7}$$

The Jacobian Matrix is defined as (8):

$$\begin{aligned} J &= (\tfrac{\partial M}{\partial q_1}, \tfrac{\partial M}{\partial q_2}, \ \ldots \ , \tfrac{\partial M}{\partial q_7}) \\ &= \begin{bmatrix} \frac{\partial M_1}{\partial q_1}, & \frac{\partial M_1}{\partial q_2}, & \cdots & \frac{\partial M_1}{\partial q_7} \\ \frac{\partial M_2}{\partial q_1}, & \frac{\partial M_2}{\partial q_2}, & \cdots & \frac{\partial M_2}{\partial q_7} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial M_6}{\partial q_1}, & \frac{\partial M_6}{\partial q_2}, & \cdots & \frac{\partial M_6}{\partial q_7} \end{bmatrix} \end{aligned} \tag{8}$$

In our expremental environment, the Jacobian matrix should be a 6×7 matrix for 6 pose variables and 7 joint angle variables, but which is not the same with all redundant manipulator. Then we get the relation between X and S in differential way:

$$\dot{X} = J\dot{S} = J(\dot{q_1}, \dot{q_2}, \ldots, \dot{q_7}) \tag{9}$$

We can't use the direct inverse matrix of J to do inverse kinematics, but can do it with the generalized inverse matrix of J (J^+) defined like this:

$$J^+ = J^T(JJ^T)^{-1} \tag{10}$$

In generalized form, we have:

$$\dot{S} = J^+\dot{X} \tag{11}$$

$$S - S_0 = \Delta S = J^+\Delta X = J^+(X - X_0) \tag{12}$$

In this way, when knowing the initial pose X_0, very near pose X, and the initial S_0, we can compute the status S relative to X according to (11)(12), which is our inverse kinematics based on, both for MLG and Newton-Raphson method we will introduce later.

III. RRT WITH EXTENDED STEPS

Since we use modified RRT to do obstacle-avoiding path planning, we can have a look at the initial RRT method to better understand the results. In this chapter, we introduce RRT and relevant algorithm in a more detail way.

A. Pre-define before RRT

We define the relevant collection and functions below of RRT before looking at its sudocode. Remind the basic problem RRT do with first: how can we run the manipulator from the initial position (knowing X_0 & S_0) to the goal position (only knowing pose of end position, taken down as X_g) without collision into the obstacle in the space? The following collection are defined to express RRT in convenience:

- A_0: whole pose space which can be reached by robot manipulator in configuration space.
- A_{free}: free space belong to A_0, i.e., $A_{free} \in A_0$, and there is no obstacle in A_{free}.
- A_{obs}: space of obstacle, i.e., $A_{obs} = A_0 \backslash A_{free}$.
- $B_0, B_{free}, Bobs$: these are the corresponding collection with A, but in the status space. The subscript takes the same meaning as in A.

Below define the main functions used in RRT. Before we focus on these functions, we should remind that RRT method will build a tree when it runs, we called it RRT tree, writen as tree T, with the initial point X_0 for pose and S_0 for status as the first tree point. We note that: $T = (V, E)$, while V is for all points and E is the relation between these points (father point and its child). Note that T is a tree, so each point has no more than one child.

- sample: generate a random point X_{ram} in pose format in A_{free}.
- distance: compute the distance of two points in A_0. Within our work, this distance is defined as the Euclid distance.
- nearest_neighbor: given the point X, this function return the nearest point X_{near} in tree T.
- steer: given the random point X_{ram} and X_{near}, this function return a new point X_{new} in A_{free}, whihc is nearer to X_{ram} than X_{near}.
- collisionTest: given point X, return 1 for obstacle free and 0 for obstacle collision.

B. RRT and its relative functions

Then we can have a look at the basic algorithm of RRT. We write it in pseudocode show in Fig.2:

Though simple as it is, RRT work well with obstacle-avoiding task. Here, we focus on the detail of the basic functions specially, for they will be related later in this paper. As defined before, we use Euclid distance but make a little change, named rrtDistance, that is defined bellow with (13):

$$rrtDistance(X_1, X_2) = \alpha||P_1 - P_2|| + (1 - \alpha)||\gamma_1 - \gamma_2||, \quad 0 < \alpha < 1. \tag{13}$$

Since we concentrate more on the accuracy of position, we mainly choose $\alpha > 0.5$, like $\alpha = 0.8$, which is use in our program. The sample function generate a point in A_0, we use X_{min} and X_{max} to limit area of A_0. Then the random point X_{ram} that sample function produce will be like (14):

```
RRT(S_0, A_0)
1. V ← {S_0}; E ← ∅; i ← 0;
2. while i < N do
3. {
4.     T ← (V, E);
5.     X_rand ← sample(i); i ← i + 1;
6.     T ← extend(T, X_rand);
7. }

extend(T, X_rand)
1. X_near = nearest_neighbor(T, X_rand);
2. X_new = steer(X_near, X_rand);
3. if X_new and collisionTest(X_new)
4.     V ← V ∪ {X_new};
5.     E ← E ∪ {(X_near, X_new)};
5.     return success;
6. else
7.     return fail;
```

Fig. 2. pseudocode of RRT algorithm.

$$X_{ram} = X_{min} + t \times (X_{max} - X_{min}), 0 < t < 1; \quad (14)$$

When t is randomly produce in interval (0, 1), sample method would produce all the points of A_0. For steer function, a new point X_{new} will be produce depending on X_{near} and X_{ram}, which works like (15):

$$X_{new} = X_{near} + s(X_{ram} - X_{near})/||X_{ram} - X_{near}||, \quad 0 < k < ||X_{ram} - X_{near}||. \quad (15)$$

We call s in (15) the step-length. In RRT algorithm, each step when steer function run, the RRT tree T will attend to explore forward s distance in A_0. Cause X_{rand} will be anywhere in the whole space, it is easy to infer that the direction tree T extends is all-dimension, i.e., no certain direction, but different direction with different time, reminded that is important cause we will discuss it later.

C. modified RRT algorithm

Though RRT can explore the whole space A_0 to get a obstacle free path to reach the goal point, it still has some shortage when it actually run. Since RRT extends only one step (with step-length s) further, it seems slow sometimes when A_0 is big enough. On the other point, RRT tree are built beginning with the initial point, leaving out the goal point in the tree-built process. Still, there are also other insufficients with RRT, which results in many modified RRT method. Here, we present bi-RRT as an example. The detail bi-RRT algorithm can be seen in [12]. We just take it in brief here.

As the shortage we tell, bi-RRT solve it by extend more steps until in collision with obstacle or reach the random point X_{ram}. And bi-RRT maintain two tree in its memory, one starting extend function from the initial point and the other from the goal point. The two tree extend to each other each tiem when extend method run until they meet in the space, which is the reason the name 'bi-RRT' come from. With such modification, bi-RRT take in count of the intial point and goal point together with a faster extending speed.

In our exprement, we just take the more steps extending thought into our code. Our multi-extend algorithm is like in Fig.3:

```
multi_extend(T, X_rand)
1. [X_new, result] = extend(T, X_rand);
2. if result = fail or collisionTest(X_new) = 0
3.     return fail;
4. while result = success
5. {
6.     [X_new, result] = extend(X_new, X_rand);
7.     if collisionTest(X_new) = 0 or
                surpass(X_new, X_rand)
8.         break;
9. }
10. return success.
```

Fig. 3. multi-step extend pseudocode

The function '$surpass(X_{new}, X_{rand})$' detect that if X_{new} is near enough to X_{rand}. The 'true' return for this function means the two points are close enough, then 'multi_extend' method will be stop, and the 'false' return will continue the extending process. With these change, if the extend direction is correct, we could get faster to the goal point, but simutaneously, we would also extend further to a bad direction, which would take up more time if this situation happens frequently. We solve this problem with RRT-GD (RRT with goal-direction), which will be related in the later paragraph. But within our process now, we have two extend functions: extend and multi-extend, and we can think of the inverse kinematics optimization, which will be discuss first.

IV. Inverse Kinematics with Newton-Raphson method

We have related some inverse kinematics method before, e.g., GP (gradient projection method), WLN (weighted least norm method), MLG (Method of Lowest Gradient)[6]. Within our program, we take Newton-Raphson method into account, for its high accuracy and speed.

A. Newton-Raphson method

We reminded back to our inverse kinematics problem: we know S_0 (and so X_0 can be compute) and X_g, how could we compute S_g? In Newton-Raphson method, which is an iteration method, each time we just go straight from the current point to goal point. We can see the iteration equation in (16):

$$S_{n+1} = S_n + J^+ \times minus(X_g, X_n). \quad (16)$$

The current status and pose are written as S_n and X_n, and J^+ is also relative to the current point, which can be compute in (10). Then when X_n is close enough to X_g, we get the answer S_n as result S_g. Note that though the function name

called 'minus', it dost just not simple minus as $X_g - X_n$, because the azimuth doesn't supplement the minus operation. We should use the Rotate-By-1-Axis method to get the corret answer, since the combination of fix-axis rotation of rigid body can be seen as one-time fix-axis.

After this work, we take down the Newton-Raphson method as Fig. 4:

$Newton\_Raphson(S_0, X_g)$
1. $compute\ X_0\ and\ J^+; i = 0; times = 0;$
2. $while\ \epsilon < rrtDistance(X_i, X_g)\ and\ times < N$
3. {
4. $\quad S_{i+1} = S_i + J^+ \times minus(X_g, X_i);$
5. $\quad i = i + 1; times = times + 1;$
6. $\quad compute\ X_i\ and\ J^+;$
7. }

Fig. 4. multi-step extend pseudocode

On Fig.4, the first and 6th line use (6) and (10) to compute it. Here we use N (we set it to 10 in program) to limit the iteration time, for the fact that Newton-Raphson method would always get the correct answer in less than 10 iteration times with accuracy of micrometer, otherwise it would fail with bias greater than ϵ, i.e., more iteration times have no use cause it can't improve the accuracy.

B. Figures and Tables

Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation Fig. 1, even at the beginning of a sentence.

TABLE II
AN EXAMPLE OF A TABLE

One	Two
Three	Four

We suggest that you use a text box to insert a graphic (which is ideally a 300 dpi TIFF or EPS file, with all fonts embedded) because, in an document, this method is somewhat more stable than directly inserting a picture.

Fig. 5. Inductance of oscillation winding on amorphous magnetic core versus DC bias magnetic field

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity Magnetization, or Magnetization, M, not just M. If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write Magnetization (A/m) or Magnetization A[m(1)], not just A/m. Do not label axes with a ratio of quantities and units. For example, write Temperature (K), not Temperature/K.

V. CONCLUSIONS

A conclusion section is not required. Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions.

APPENDIX

Appendixes should appear before the acknowledgment.

ACKNOWLEDGMENT

The preferred spelling of the word acknowledgment in America is without an e after the g. Avoid the stilted expression, One of us (R. B. G.) thanks . . . Instead, try R. B. G. thanks. Put sponsor acknowledgments in the unnumbered footnote on the first page.

References are important to the reader; therefore, each citation must be complete and correct. If at all possible, references should be commonly available publications.

REFERENCES

[1] G. O. Young, Synthetic structure of industrial plastics (Book style with paper title and editor), in Plastics, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 1564.
[2] W.-K. Chen, Linear Networks and Systems (Book style). Belmont, CA: Wadsworth, 1993, pp. 123135.
[3] H. Poor, An Introduction to Signal Detection and Estimation. New York: Springer-Verlag, 1985, ch. 4.
[4] B. Smith, An approach to graphs of linear forms (Unpublished work style), unpublished.
[5] E. H. Miller, A note on reflector arrays (Periodical styleAccepted for publication), IEEE Trans. Antennas Propagat., to be publised.
[6] J. Wang, Fundamentals of erbium-doped fiber amplifiers arrays (Periodical styleSubmitted for publication), IEEE J. Quantum Electron., submitted for publication.
[7] C. J. Kaufman, Rocky Mountain Research Lab., Boulder, CO, private communication, May 1995.
[8] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, Electron spectroscopy studies on magneto-optical media and plastic substrate interfaces(Translation Journals style), IEEE Transl. J. Magn.Jpn., vol. 2, Aug. 1987, pp. 740741 [Dig. 9th Annu. Conf. Magnetics Japan, 1982, p. 301].
[9] M. Young, The Techincal Writers Handbook. Mill Valley, CA: University Science, 1989.
[10] J. U. Duncombe, Infrared navigationPart I: An assessment of feasibility (Periodical style), IEEE Trans. Electron Devices, vol. ED-11, pp. 3439, Jan. 1959.
[11] S. Chen, B. Mulgrew, and P. M. Grant, A clustering technique for digital communications channel equalization using radial basis function networks, IEEE Trans. Neural Networks, vol. 4, pp. 570578, July 1993.
[12] R. W. Lucky, Automatic equalization for digital communication, Bell Syst. Tech. J., vol. 44, no. 4, pp. 547588, Apr. 1965.
[13] S. P. Bingulac, On the compatibility of adaptive controllers (Published Conference Proceedings style), in Proc. 4th Annu. Allerton Conf. Circuits and Systems Theory, New York, 1994, pp. 816.
[14] G. R. Faulhaber, Design of service systems with priority reservation, in Conf. Rec. 1995 IEEE Int. Conf. Communications, pp. 38.
[15] W. D. Doyle, Magnetization reversal in films with biaxial anisotropy, in 1987 Proc. INTERMAG Conf., pp. 2.2-12.2-6.

[16] G. W. Juette and L. E. Zeffanella, Radio noise currents n short sections on bundle conductors (Presented Conference Paper style), presented at the IEEE Summer power Meeting, Dallas, TX, June 2227, 1990, Paper 90 SM 690-0 PWRS.
[17] J. G. Kreifeldt, An analysis of surface-detected EMG as an amplitude-modulated noise, presented at the 1989 Int. Conf. Medicine and Biological Engineering, Chicago, IL.
[18] J. Williams, Narrow-band analyzer (Thesis or Dissertation style), Ph.D. dissertation, Dept. Elect. Eng., Harvard Univ., Cambridge, MA, 1993.
[19] N. Kawasaki, Parametric study of thermal and chemical nonequilibrium nozzle flow, M.S. thesis, Dept. Electron. Eng., Osaka Univ., Osaka, Japan, 1993.
[20] J. P. Wilkinson, Nonlinear resonant circuit devices (Patent style), U.S. Patent 3 624 12, July 16, 1990.