Collision-Avoidance Problem Solution of Robot Manipulators Via CGA

Othman MK. Alsmadi[1] and Za'er S. Abo-Hammour[2]

[1]Department of Electrical Engineering, University of Jordan, Amman, Jordan
othman_mk@yahoo.com
[2]Department of Mechatronics Engineering, University of Jordan, Amman, Jordan
zaer_hr@yahoo.com

Abstract

In this paper, continuous genetic algorithms (CGAs) along with distance algorithm are used to solve the path planning generating problem for collisions-free robot manipulators. For the distance algorithm, built upon especial models for obstacles and manipulator link, a spherical and cylispherical models were chosen for obstacles and links respectively. Based on the distance between the obstacle and the link, a robot collisions-free motion was generated. In order to obtain the desired path, the CGAs were employed. Simulation results show that the combined distance algorithm and CGAs provide the robot with the desired path along with obstacle avoidance. This algorithm has been applied to the PUMA 560.

1. Introduction

Over the last decades, robot manipulators have received much of the research community attention due to their wide applications in industry for many tasks [1,2]. Obstacle avoidance is one of the key technology areas that must be developed in order to allow the application of robotics to continue to grow and for robots to operate effectively in cluttered environments.

Different approaches have been used to plan manipulator motion. For the optimum motion planning, it is well know that it is vital in industry since it improves the productivity and the precision of robot systems [2,3]. The precision of the process (quality) may be improved at the programming level by defining the Cartesian path with more path points and employing a more sophisticated path planning method, while the production rate (productivity) is improved by incorporating an efficient trajectory generation algorithm which results in time-optimal trajectories and correspondingly increases the production rate.

A great deal of work has been devoted to solve the motion planning problem, as extensively seen in [2,4]. However, many weaknesses have been observed as investigations took a place [5]. Genetic algorithms, on the other hand, have received a considerable attention in robotics [6,7,8]. However, generally they could not fully exploit the abilities of GAs and has the different drawbacks [3]. To overcome such problems, we used Continuous Genetic Algorithms (CGAs), which were developed by Abo-Hummour [9] as an efficient method for the solution of optimization problems in which the parameters to be optimized are correlated with each other or the smoothness of the solution curve must be achieved. CGAs have been successfully applied in the motion planning of robot manipulators [2] and in the numerical solution of two-point boundary value problems [10]. Also, they have been applied in the solution of fuzzy differential equations [11]. For more details on the CGAs including their justification for use, conditions on smoothness of the functions used in the algorithms, etc., please refer to [2,10].

In this paper, we extend the work of [10] to solve the path generation problem in the case of stationary obstacles in the workspace. Geometric representations of the obstacles and robot-obstacle distance measurements are incorporated in the CGA in order to generate an accurate, collision-free path for the manipulator.

2. Modeling and Distance Calculation

There has been a lot of obstacle avoidance research that uses simple, often smooth modeling primitives. Perry and Tesar [12] chose to use three different obstacle models. Wherever it was practical, simple shapes (spheres or cylispheres) were used to model obstacles. More complex shapes were modeled using superquadric surfaces. In this paper, manipulators are modeled using cylispheres (i.e. cylinders with hemispherical ends) and obstacles are modeled using spheres. The sphere, as shown in Figure 1, is the simplest shape that can be used to model a 3D object. The symmetric properties of a sphere eliminate all orientation issues. Two pieces of information completely specify the location of a sphere. These are the center point and the radius.

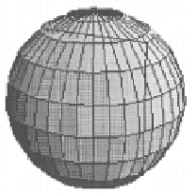


Fig. 1. Sphere model [4].

To successfully determine the minimum distance between a manipulator and an obstacle, one must model the manipulator as well as the obstacle. The cylisphere, as shown in Figure 2, is a natural extension of a sphere. It is a cylinder with hemispheres at each end, and is symmetrical about its 'long' axis. Three pieces of information completely specify the location and orientation of a cylisphere. These are the two endpoints and the radius.

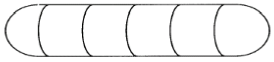


Fig. 2. Cylisphere model [4].

The locations of the cylispheres used to model a manipulator are determined based on the forward kinematics for that manipulator. The forward kinematics provides transformation matrices from predetermined points on the manipulator to the global coordinate frame. The endpoints for each cylisphere have a fixed offset from one of these predetermined points. Therefore, the endpoints of the cylispheres move as the transformation matrices change due to manipulator motion. A more detailed description is found in [13].

For obstacle avoidance, it is extremely important to always know the minimum distance between a manipulator and all the surrounding obstacles. In their research, Perry and Tesar [12] developed functions for determining minimum distances between a cylisphere and a sphere. Figure 3 shows an illustrative graph used to derive the minimum distance between a cylisphere and a sphere. In this figure, Line 1 is a cylisphere with a zero radius, and point P_3 is a sphere with a zero radius. The shortest line, Line 2, between Line 1 and point P_3 is perpendicular to Line 1. Point P_4 is the only point on both lines. The location of point P_4 can be found by representing both lines parametrically and then solving for the location of point P_4 in terms of the location of the other three points: P_1, P_2, and P_3. As a result, it can be easily shown that the minimum distance between the cylisphere and the sphere is

$$D_m = \sqrt{(x_4 - x_3)^2 + (y_4 - y_3)^2 + (z_4 - z_3)^2} \tag{1}$$

For a cylisphere and a sphere that both have a non-zero radius, the minimum distance is calculated as discussed above, then the radii of both cylisphere and sphere (RL and RO, respectively) are subtracted from the zero-radius minimum distance to give the true minimum distance, $D_{\min}$, as shown in the following equation

$$D_{\min} = D_m - (R_L + R_0) \tag{2}$$

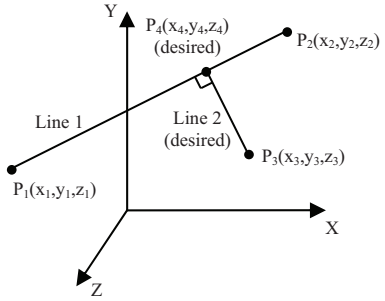


Fig. 3. Distance derivation between the sphere and cylisphere.

3. Formulation of the Path Generation Problem

Let us consider a robot manipulator with *M* degrees of mobility and *N* task space coordinates. Assume that a desired Cartesian path, P_{dc}, is given, and the problem is to find the set of joint paths, P_θ, such that the accumulative deviation between the generated Cartesian path, P_{gc}, and the desired Cartesian path, P_{dc}, is minimum. To do that, the desired geometric Cartesian path, in our approach, is sampled. The number of sampling points (path points or knots) is specified by the programmer and depends on the desired accuracy of the generated path. The accuracy of the generated path increases as the number of path points increases. However, a limiting case for this number is the path update rate; that is, we can increase the number of path points until we reach the limit using the following equation

$$N_K = T_t P_{ur} \tag{3}$$

where N_k is the number of knots along the geometric path, T_t is the total traveling time from the starting configuration to the final configuration along the desired path, and P_{ur} is the path update rate of the manipulator.

After sampling the geometric path at the path update rate for best accuracy, the generated values of the joint angles using the continuous genetic algorithm, P_θ, are used by the direct (forward) kinematics model of the robot to obtain the generated Cartesian path given by

$$P_{gc} = F_K(P_\theta), \tag{4}$$

where F_k represents the robot forward kinematics model.

The deviation between the desired Cartesian path, P_{dc}, and the generated Cartesian path, P_{gc}, at some general path point, *i*, is given as

$$E(i) = \sum_{K=1}^{N} \left| P_{dc}(k,i) - P_{gc}(k,i) \right|. \tag{5}$$

where the accumulative deviation between the two paths is, therefore, given by

$$E = \sum_{i=1}^{N_k} \sum_{k=1}^{N} \left| P_{dc}(k,i) - P_{gc}(k,i) \right| = \sum_{i=1}^{N_k} E(i) \tag{6}$$

The fitness function developed, a nonnegative measure of the quality of individuals, is defined as [14]

$$F = \frac{1}{1 + E + P} \tag{7}$$

where *P* is the penalty function introduced so that the fitness of the solution depends on both its closeness to the desired path as well as the distance from the obstacles. This penalty is given by

$$P = P_{\min} + w_{D\min} P_{D\min} + w_{\overline{D}} P_{\overline{D}} + w_{NC} P_{NC} \tag{8}$$

where $P_{\min}$ is the minimum value of P, $w_{D\min}$, $w_{\overline{D}}$, and w_{NC} are weight values and

$$P_{D\min} = \frac{\varepsilon - D_{\min All}}{\varepsilon}, \; P_{\overline{D}} = \frac{\varepsilon - \overline{D}_{\min}}{\varepsilon}, \; P_{NC} = \frac{N_{col}}{N_s} \tag{9}$$

where ε is a defined threshold, $D_{\min All}$ is the least value of $D_{\min}$ encountered throughout the path, $\overline{D}_{\min}$ is the average of all minimum distances through the path, N_{col} is the number of collisions, and N_s is the number of steps. The optimal solution

of the problem is obtained when the deviation function, *E*, approaches zero, and the penalty function, *P*, also approaches zero. Correspondingly, the fitness function, *F*, approaches unity. As a result, the path generation problem is formulated as minimization problems of the deviation and penalty functions or as a maximization problem of the fitness function.

4. Continuous Genetic Algorithms (CGAs)

Genetic Algorithms (GAs) were developed by John Holland [14] and are based on principles inspired from the genetic and evolution mechanisms observed in natural systems. Their basic principle is the maintenance of a population of solutions to the problem that evolves towards the global optimum. CGAs, an extension to the CA, use smooth operators and avoid sharp jumps in the parameter value. CGAs have the same steps as in a typical genetic algorithm; the steps of the continuous genetic algorithm used in this paper are as follows (for more details, please refer to [14]):

1) ***Initialization*:** Randomly generating an initial population comprising N_p smooth individuals. Two smooth functions are proposed for initializing the population: the modified Gaussian function and the tangent hyperbolic function [14]. The modified Gaussian function is given as follows:

$$P_J(h,i) = \theta_{initial}(h) + \left(\frac{\theta_{final}(h) - \theta_{initial}(h)}{N_K - 1}\right) \times (i-1) + A\exp\left(\frac{-(i-\mu)^2}{2\sigma^2}\right) \quad (10)$$

while the tangent hyperbolic function is governed by the equation:

$$P_j(h,i) = \theta_{initial}(h) + (\theta_{final}(h) - \theta_{initial}(h)) \times 0.5\left(1 + \tanh\left[\frac{i-\mu}{\sigma}\right]\right) \quad (11)$$

for all $1 \le h \le M$ and $1 \le i \le N_k$, where $P_j(h,i)$ is the ith path point angle of the hth joint for the jth parent, $\theta_{initial}(h)$ is the initial angle of the hth joint at the initial configuration of the end-effector, $\theta_{final}(h)$ is the final angle of the hth joint at the final configuration of the end-effector, N_k is the total number of knots (sampling points) across the Cartesian path, A represents a random number within the range $[-3R(h), 3R(h)]$, where $R(h) = \left|\theta_{final}(h) - \theta_{initial}(h)\right|$, μ is a random number within the range $[1, N_k]$, and σ is a random number within the range $[1, N_k/6]$. The difference between these two functions lies in the fact that the modified Gaussian function results in an overshoot or an under shoot, while the tangent hyperbolic function does not result in either. For both functions, μ specifies the center of the function, while σ specifies its degree of dispersion. The two initialization functions are shown in Figure 4.

2) ***Evaluation*:** A nonnegative measure of quality (fitness), used to reflect the degree of goodness of the individual, is calculated for each individual in the population as given in (7) which is based on (8) and (9) as well.

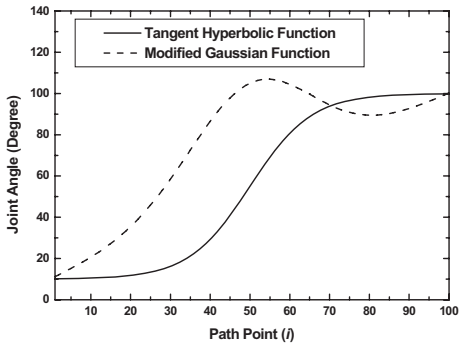


Fig. 4. Initialization functions.

3) ***Selection*:** In the selection process, the best individuals receive more copies in subsequent generations so that their desirable traits may be passed onto their offspring. This step insures that the overall quality of the population increases from one generation to the next. Due to the probabilistic nature of selection, individuals can merely be expected, but not guaranteed, to reproduce in proportion to their fitness.

4) ***Crossover:*** Crossover provides the means by which valuable information is shared among the population. It combines the features of two parent individuals, i.e., j and k, to form two children individuals, i.e., L and L+1, that may have new patterns compared to those of their parents, and plays a central role in GAs. The crossover process in our algorithm is expressed as

$$C_L = W(h,i)P(h,i) + (1 - W(h,i))P_k(h,i) \quad (12)$$

$$C_{L+1} = (1 - W(h,i))P(h,i) + W(h,i)P_k(h,i) \quad (13)$$

$$W(h,i) = 0.5\left(1 + \tanh\left[\frac{i-\mu}{\sigma}\right]\right) \quad (14)$$

for all $1 \le h \le M$ and $1 \le i \le N_k$, where P_j and P_k represent the two parents chosen from the mating pool, C_L and C_{L+1} are the two children obtained through the crossover process, W represents the crossover weighting function within the range [0, 1], μ is a random number within the range $[1, N_k]$, and σ is a random number within the range $[1, N_v/6]$.

5) ***Mutation*:** Mutation is often introduced to guard against premature convergence. Generally, over a period of several generations, the gene pool tends to become more and more homogeneous. The purpose of mutation is to introduce occasional perturbations to the parameters to maintain genetic diversity within the population. The mutation process in our algorithm is governed by:

$$C_j^m(h,i) = C_j(h,i) + dM(h,i) \quad (15)$$

$$M(h,i) = \exp\left(\frac{-(i-\mu)^2}{2\sigma^2}\right) \quad (16)$$

where C_j represents the jth child produced through the crossover process, C_j^m is the mutated jth child, M is the

Gaussian mutation function, *d* represents a random number within the range [*-range(h), range(h)*], with *range(h)* representing the difference between the minimum and maximum values of the child C_j, and μ and σ are as given in the crossover process.

In the mutation process, each individual child undergoes mutation with probability P_{mi}. However, for each child that should undergo a mutation process, individual joints are mutated with probability P_{mj}. If the P_{mi} value is set to 0.5 and the P_{mj} value is set to 0.5, then one child out of two children is likely to be mutated, and within that child, $M/2$ of the joints' paths are likely to be mutated.

6) ***Replacement*:** After generating the offspring's population through the application of the genetic operators to the parents` population, the parents` population is totally or partially replaced by the offspring's population. This completes the "life cycle" of the population.

7) ***Termination*:** The GA is terminated when some convergence criterion is met. Possible convergence criteria are: the fitness of the best individual so far found exceeds a threshold value, the maximum number of generations is reached, or the progress limit (the improvement in the fitness value of the best member of the population over a specified number of generations being less than some predefined threshold) is reached. After terminating the algorithm, the optimal solution of the problem is the best individual so far found.

Based on the Continuous Genetic Algorithms CGAs, which use smooth operators and avoid sharp jumps in the parameter value, the fitness function will be used for the path planning of the robot manipulator.

5. Simulation Results

As an illustrative example, the algorithm will be implemented for the following desired path and obstacle information:

$$x_{initial} = 0.1,\ x_{final} = 0.1 + \frac{1}{8}\pi.$$

$$P_{dc}(1,i) = X_{dc}(i) = x_{initial} + \frac{x_{final} - x_{initial}}{N_k - 1}(i-1) \qquad (17)$$

$$P_{dc}(2,i) = Y_{dc}(i) = 0.25 + 0.25\sin\left[8(1,i) - x_{initial})\right]$$

$$P_{dc}(3,i) = Z_{dc}(i) = 0, \qquad 1 \le i \le N_k$$

In addition to that, there are two obstacles in space (x, y, z) at the following locations: Obstacle_1 = (-0.2, 0.46, 0)cm; Obstacle_2 = (0.1, -0.57, 0)cm. The radii of the obstacles are 5 cm for Obstacle_1 and 7 cm for Obstacle–2.

The algorithm is used to solve the path generation problem of the PUMA manipulators, in which the link radius is assumed to be zero. For this case, *N*=3, *M*=3, and the forward kinematics model is given in [15]. For this manipulator, the z-coordinates are kept at zero in the first path but have a value in the second. This degree of freedom is not taken into consideration for the planar case. The number of path points, N_k, is 100 points.

In the simulation results, Figure 5 and Figure 6 show the first and second feasible solutions, respectively, for the PUMA manipulator in the given path. It can be clearly seen that the solutions are obstacle free. Figure 7 shows the evolutionary progress plots of the best-fitness individual. As can be seen, the algorithm takes about 80 generations to reach a near-optimal solution with a fitness value of 0.8, while it takes about 170 generations to reach the optimal solution from the near-optimal one. This shows that most of the computational burden of the algorithm is spent in the fine-tuning stage whereas the near-optimal solutions are found very early in the course of the CGA.

Figure 8 shows the deviations, in the generated path obtained using the CGA, from the desired ones. It is clear that the average deviation does not exceed 0.0007m for the PUMA configuration, which proves the effectiveness of the CGA in generating paths with minimum deviations from the desired ones. In addition to that, the generated path does not pass through obstacle positions and thus it is a collision-free type.

The results shown in Figures 7 and 8 relate to the given path as given in Equation (15), which has no collisions for all the solutions. However, the chosen solution will be determined based on the maximum fitness as well as the penalty function.

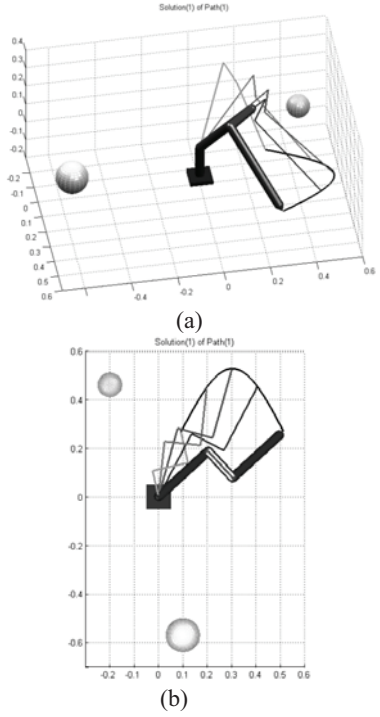
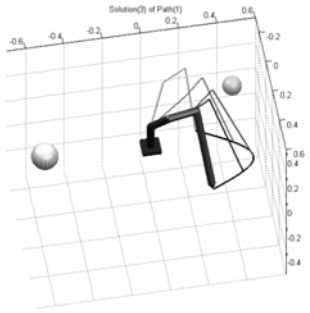


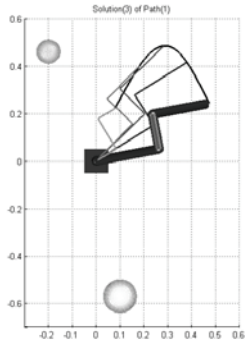
Fig. 5. First feasible solution.

6. Conclusion

The CGAs along with the geometry approach have been applied as a path planner for avoiding stationary obstacles in the manipulator's workspace. For avoiding obstacles using the geometry approach, it is important to always know the minimum distance between the manipulator links and all the surrounding obstacles. This will include modeling of obstacles and manipulator's links in the workspace where obstacles are modeled as spheres, while links are modeled as cylisphere. The locations of the cylispheres, used to model a manipulator's links, are determined based on the forward kinematics for that manipulator. As a result of the presented research, it was found that the genetic algorithms can be used as an efficient tool in robotics application, which in our case has two aims; first, obtaining a path with a minimum deviation from the desired one; second, in the case of finding several paths, these paths will be reduced when finding obstacle(s) in their ways.



(a)



(b)

Fig. 6. Second feasible solution.

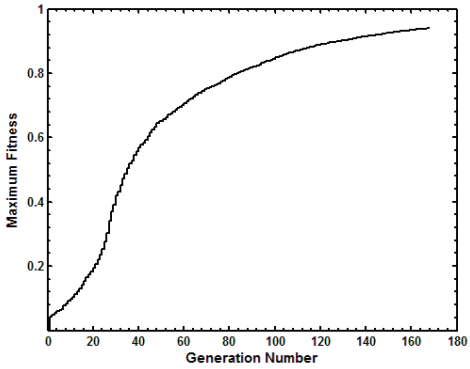


Fig. 7. Evolutionary progress for the best individual

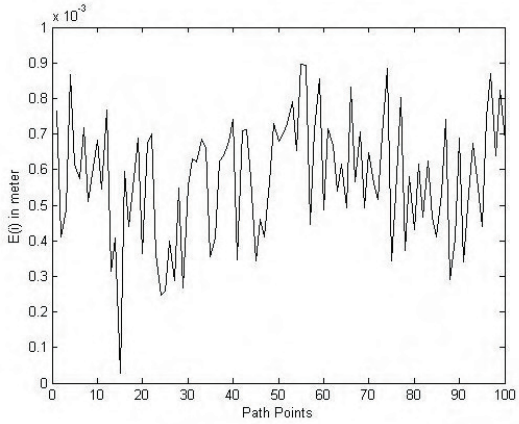


Fig. 8. Path point deviations.

7. References

[1] C. Belta and V. Kumar, "Motion Generation for Formation of Robots: a Geometric Approach", *Robotics and Automation*, vol. 18, no. 3, pp. 47-61, 2001.

[2] Z.S. Abo-Hammour, P. Ji, X-J. Liu, A. S. Morris, C-W. Park, N-C. Park, B. Subudhi, A.A. Tseng, J. Wang, H. Wu, H-S. Yang, and S. Yildirim, "Robot Manipulators: New Research", Nova Science Publishers, Inc. New York, 2005.

[3] K. Sugihara, "Genetic algorithms for adaptive planning of path and trajectory of a mobile robot in 2D terrains", *IEICE Trans. Inf. & Syst., E82-D(1)*, pp. 309-317, 1999.

[4] R.P. Paul, "Manipulator Cartesian path control", *IEEE Transactions on Systems*, Man and Cybernetics, vol. 9, no. 11, pp. 702-711, 1979.

[5] K.S. Fu, R.C. Gonzalez, C.S.G. Lee, "Robotics: control, sensing, vision, and intelligence", McGraw-Hill, New York, 1987.

[6] G. Zhang, L. Gao, X. Li, and P. Li, "Variable Neighborhood Genetic Algorithm for the Flexible Job Shop Scheduling Problems", *ICIRA 2008*, Part II Springer-Verlag, Berlin Heidelberg, 2008.

[7] O. Castillo, L. Trujillo, P. Melin, "Multiple objective genetic algorithms for path-planning optimization in autonomous mobile robots", *Soft Computing,* vol. 11, pp. 269–279, 2007.

[8] F. Gutierrez, J. Atkinson, "Autonomous Robotics Self-Localization Using Genetic Algorithms", *(ICTAI '09) 21st International Conference on Tools with Artificial Intelligence,* Newark, New Jersey, Nov. 2-5, 2009.

[9] Z.S. Abo-Hammour, "Advanced continuous genetic algorithms and their applications in the motion planning of robotic manipulators and the numerical solution of boundary value problems", Ph.D. Thesis, Quiad-Azam University, Pakistan, 2002.

[10] Z.S. Abo-Hammour, M. Yusuf, N.M. Mirza, S.M. Mirza, M. Arif, "Numerical solution of second-order, two-point boundary value problems using continuous genetic algorithms", *International Journal for Numerical Methods in engineering*, vol. 61, pp. 1219-1242, 2004.

[11] O.M. Abo-Arqoub, "Numerical Solution of Fuzzy Differential Equations Using Continuous Genetic Algorithms", Ph.D. thesis, University of Jordan, Amman, Jordan, 2008.

[12] B.R. Perry and D. Tesar, "The Development of Distance Functions and Their Higher-Order Properties for Artificial Potential Field-Based Obstacle Avoidance", M.S. thesis, The University of Texas at Austin, Austin, TS, 1995.

[13] J. Craig, "Introduction to Robotics: Mechanics and Control", Addison-Wesley, New York , 1989.

[14] Z.S. Abo-Hammour, N. Mirza, S. Mirza, M. Arif, "Cartesian Path Generation of Robot Manipulators Using Continuous Genetic Algorithms", *Robotics and Autonomous Systems*, vol. 41, no. 40, pp. 179-223, 2002.

[15] K.S. Fu and R.C. Gonzalez, C.S.G. Lee, "Robotics: Control, Sensing, Vision, and Intelligence", McGraw-Hill, New York, 1987.