Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 11, November-2019

Emergent Behavior based Limb Control using the Repulsive Root Technique

Patrick McDowell, Kuo-Pao Yang
Department of Computer Science
Southeastern Louisiana University
Hammond, LA 70402 USA

***Abstract*— The goal of this work was to create a method of controlling multiple joint arms that is adaptable to varying conditions over time. The method uses a combination of techniques; initially it uses emergent behavior, each segment acting independently, from which the functionality of the arm emerges. In order to reach targets at all points within the reach of the limb a "repulsive root" technique was employed. This paper details algebraic forms of solving for the limb segment angles for 2 and 3 degree of freedom arms, references the Jacobian Inverse technique, and then introduces the repulsive root technique. The system was tested in simulation using C/C++ and OpenGL; results are discussed. This method is not meant to be the final solution to the problem, but instead an exploration technique that provides data to an opportunistic learning system.**

Keywords- Robot Kinematics; Emergent Behavior; Repulsive Root Technique

I. INTRODUCTION

Robotic limbs are common in both mobile robots and in assembly robots in manufacturing plants. To find the position of the end effector of the arm, forward kinematics [1] is used. To determine the joint angles needed to put the end effector in some desired position, inverse kinematics [2] is used. For limbs with 2 or 3 degrees of freedom, there are algebraic/trigonometric solutions that are fairly straight forward. For arms with more than 3 degrees the inverse jacobian [3] is used along often in combination with a stepwise numerical solution.

One of the problems that occurs with these solution methods is that as the number of joints increases, the solution gets more and more complex. Part of this is because that each new joint introduces the possibility of having alternate solutions to the problem. Fig. 1 below illustrates the problem. As can be seen, the end effector can be placed at position P_3 two different ways. For each new degree of freedom added to the limb this problem is compounded.

The following section briefly details the algebra of a 2-degree of freedom (DOF) solution. The location of P_1 will be assumed to be (0,0), and the lengths of the segments, and the angles between the segments are known. Using these values, the locations of the ends of the segments, P_2 and P_3 can be calculated using forward kinematics as is shown below.

$$P_{2x} = L_1 \cos \theta_1$$
$$P_{2y} = L_1 \sin \theta_1$$

$$P_{3x} = L_1 \cos \theta_1 + L_1 \cos (\theta_1 + \theta_2)$$
$$P_{3y} = L_1 \sin \theta_1 + L_1 \sin (\theta_1 + \theta_2)$$

The following definitions and equations describe the locations of the relevant parts of the 2 DOF arm in a inverse kinematics situation. This situation is very important because it tells what θ_1 and θ_2 need to be for a desired end effector's position (P_3).

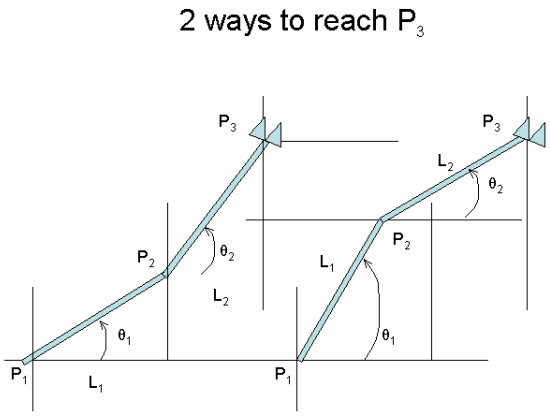

Fig. 1. This figure shows that for the arm to reach point P3 there are two different solutions.

Two important formulas that will be used to algebraically solve these equations will be the Law of Sins and Law of Cosines. They are stated below:

Law of Sins:

$$\frac{a}{\sin(A)} = \frac{b}{\sin(B)} = \frac{c}{\sin(C)}$$ [see Fig. 2 below]

$$D = \tan^{-1}\left(\frac{P_{3y} - P_{1y}}{P_{3x} - P_{1x}} \right)$$

$$c = \sqrt{(P_{3x} - P_{1x})^2 + (P_{3y} - P_{1y})^2}$$

Law of Cosines:

$$a^2 = b^2 + c^2 - 2bc \cos A$$

rearranging we get:

$$A = \cos^{-1}\left(\frac{b^2 + c^2 - a^2}{2bc} \right)$$

$$\frac{a}{\sin(A)} = \frac{c}{\sin(C)}$$

$$C = \sin^{-1}\left(\frac{\sin(A)c}{a} \right)$$

Total Angle = D + A

Or

Total Angle = D - A

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 11, November-2019

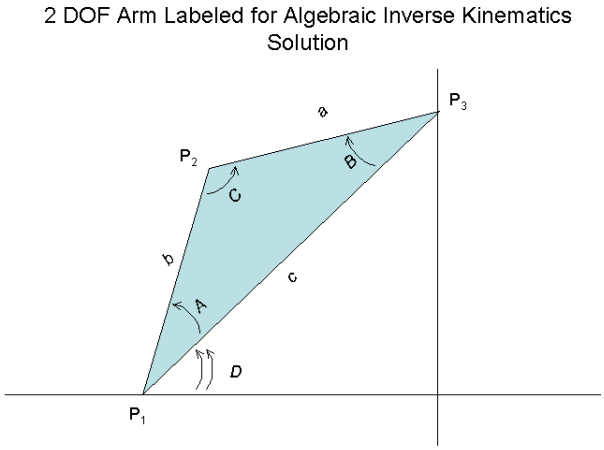

Fig. 2. This figure shows a triangle outlining the 2DOF inverse kinematic solution. Side b and c represent the robot arm segment lengths.

Algebraic solutions for the forward and inverse kinematics of a 3 DOF arm are somewhat similar to those described in the previous paragraphs. The difference is that the algebra gets more complex. In this section we show a numeric example for the 3 DOF arm, but do not show all of the details of the solution [4][5].

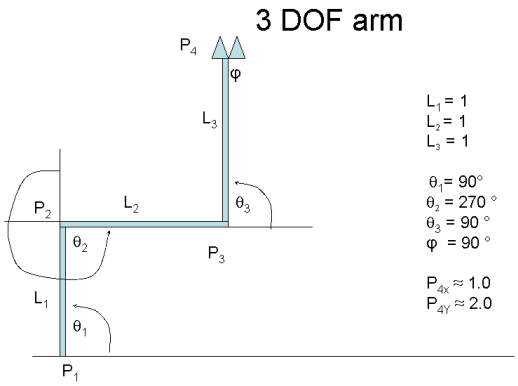

Fig. 3. This figure shows a 3 degree of freedom arm with the end effector at coordinates (1, 2).

Looking to Fig. 3, it can be seen that the end effector is located at P_4 at position (1, 2). Since the end effector is pointing straight up, $\varphi = 90^{\circ}$. In this example, we will use the given angles and segment lengths to find the position of the end effector using forward kinematics. Then we will use the position of the end effector, the angle of the end effector and the lengths of the segments to find angles between the segments, inverse kinematics.

Forward

$x = L_1\cos\theta_1 + L_2\cos(\theta_1 + \theta_2) + L_3\cos(\theta_1 + \theta_2 + \theta_3)$
$x = \cos(90) + \cos(90 + 270) + \cos(90 + 270 + 90)$
$x = 1$

$y = L_1\sin\theta_1 + L_2\sin(\theta_1 + \theta_2) + L_3\sin(\theta_1 + \theta_2 + \theta_3)$
$y = \sin(90) + \sin(90 + 270) + \sin(90 + 270 + 90)$
$y = 2$

$P_4 = (1, 2)$

Inverse

$x' = x - L_3\cos\varphi$
$x' = 1 - 1\cos(90)$
$x' = 1$

$y' = y - L_3\sin\varphi$
$y' = 2 - 1\sin(90)$
$y' = 1$

$(x', y') = (1, 1)$

$\gamma = \tan^{-1}(-y'/\sqrt{x'^2 + y'^2} \;/\; x'/\sqrt{x'^2 + y'^2})$

$\gamma = \tan^{-1}(-1/\sqrt{2} \;/\; 1/\sqrt{2})$
$\gamma = \tan^{-1}(-1)$
$\gamma = -45$

$\theta_1 = \gamma + \sigma\cos^{-1}\big(-(x'^2 + y'^2 + L_1^2 - L_2^2) / \sqrt{(-2L_1x')^2 + (-2L_1y')^2}\big)$

$\theta_1 = -45 \pm \cos^{-1}(-(1^2 + 1^2 + 1^2 - 1^2) / (2(1)\sqrt{1^2 + 1^2}))$

$\theta_1 = -45 \pm \cos^{-1}(-2/2\sqrt{2})$
$\theta_1 = -45 \pm 135$
$\theta_1 = 90 \text{ or } -180$

Select 90 based on the diagram in Fig. 3.

$\theta_1 = 90$
$\theta_2 = \tan^{-1}((y' - L_1\sin\theta_1)/L_2 \;/\; (x' - L_1\cos\theta_1)/L_2) - \theta_1$
$\theta_2 = \tan^{-1}(0/1) - 90$
$\theta_2 = 0 - 90$
$\theta_2 = -90 \text{ or } 270$

Looking at the diagram in Fig. 3, either solution will work, we select 270.

$\theta_3 = \varphi - (\theta_1 + \theta_2)$
$\theta_3 = 90 - (90 + 270)$
$\theta_3 = 90 - 360$
$\theta_3 = 90 - 0$
$\theta_3 = 90$

The above discussion has shown how to find the solution to problems in which the limb/robotic arm has 2 or 3 degrees of freedom. The forward solutions to limbs with more degrees of freedom is straight forward, as follows:

$$x = L_1\cos\theta_1 + L_2\cos(\theta_1 + \theta_2) + L_3\cos(\theta_1 + \theta_2 + \theta_3) \ldots\ldots\ldots L_n\cos(\theta_1 + \theta_2 + \theta_{n-1} + \theta_n)$$
$$y = L_1\sin\theta_1 + L_2\sin(\theta_1 + \theta_2) + L_3\sin(\theta_1 + \theta_2 + \theta_3) \ldots\ldots\ldots L_n\sin(\theta_1 + \theta_2 + \theta_{n-1} + \theta_n)$$

But, when more degrees of freedom are used in the inverse solution, a method relying on the Inverse Jacobian matrix is used [6]. While it allows for more flexibility in the numbers of degrees of freedom that it offers, it is a more complex procedure. As stated before the complexity of the matrix, and its solution grows with each added segment and joint in the limb.

The next section describes a simple emergent behavior based algorithm that iteratively find a solution to the problem [7]. It is relatively low in computational costs, and is effective.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 11, November-2019

II. APPROACH

When high numbers of segments are used in limbs, traditional control methods relying upon algebraic/Jacoby Inverse become complex and CPU intense operations. The method described here takes its inspiration from biology; the thinking is that animals with multi-segmented or highly flexible limbs, whether vertebrate or invertebrate, do not somehow do formal inverse kinematics in their brains in order to control their limbs.

Instead, it is proposed that they use a combination of memory and exploration. The thrust of this work is in exploration. Put another way, in order to effectively control a multi-jointed limb, a memory bank of positions is built using exploration. This memory bank can then be used as a basis for controlling the limb without having to do the exploration. For example, if the limb must be moved to a point that has been previously visited, the memory can be accessed based on the location, and the needed joint angles can be accessed and delivered to the limbs actuator controllers. If the point has not been visited, several approaches can be taken, including the following:

- Nearest neighbor. The limb can be moved directly to the nearest previously explored point, and then the exploration routine can be started from there.
- The memory bank can be used by a concurrently running learning system that learns by generating a function based on the contents of the memory. Neural networks are a strong candidate for this method.

The algorithm developed for this work is emergent behavior based, meaning that during exploration each limb segment acts on its own to reach the target position. Initially, stage 1 of the exploration, we let each joint act independently in order to reach the target. Each joint independently seeks out the target position using a pair of left and right proximity sensors, see Fig. 4. To move towards the target, the segment adjusts its angle towards the sensor which detects the highest target strength. The effect is that the arm will smoothly move towards the target, if, and only if the target is not close to the base of the arm.

Target positions that are close to the base of the arm require the arm to contort itself into positions that require some joints to head away from the object and some to head towards the object. In order to solve this aspect of the problem we introduce the idea of the "repulsive root".

This idea is for when the target is within the perimeter of the total length of the segments. Once the stage one emergent behavior converges, starting at the lowest segment (it's base or "root"), segments are switched to moving away from the target, instead of to it. Using this strategy, a bend in the arm can be formed, and the bend will be smooth because the lowest segments will be repulsed one by one until the target is reached. Remember that while the lower segments are moving away from the target, the upper segments will be moving towards it.

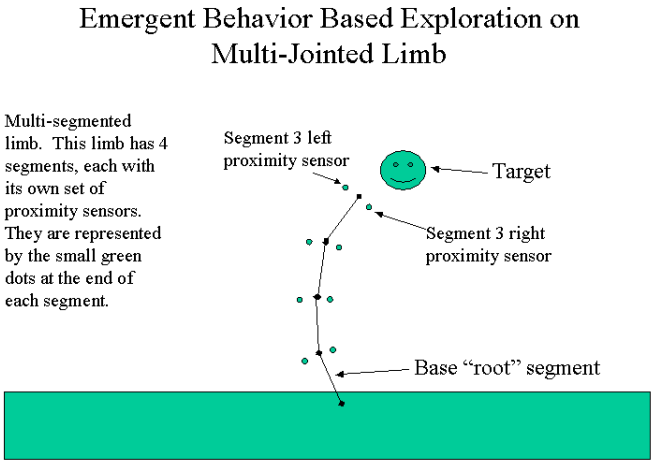

Fig. 4. This figure illustrates the basic components of the smart limb. Each segment of the limb has its own proximity sensors for detecting the target. Each segment acts in a semi-autonomous fashion in that they all try to get as close to the target as possible, with no inter-segment communication.

III. IMPLEMENTATION AND TESTING

To test the algorithm, a program was written in C/C++ and OpenGL graphics. The arms are instantiated as objects in which the number of segments is an input parameter. The control techniques tested were algebraic (using the previously discussed methods), a table based learning technique in which a two jointed arm goes through a learning stage that populates a lookup table based on location, stage1 of the emergent behavior exploration technique, and finally the emergent behavior technique paired with the repulsive root. Fig. 5 below shows a 30-link arm touching its target (the red dot).

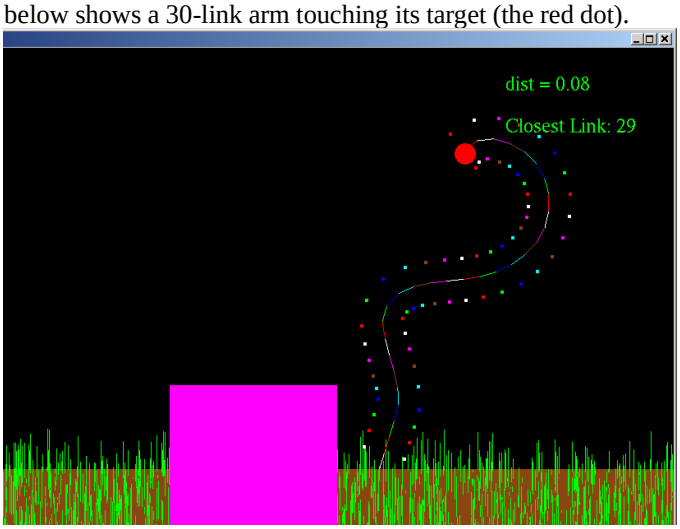

Fig. 5. This figure shows the simulation of a 30-link limb touching its target (the red dot). Note the dots on the left and right side of the multi-colored limb. These dots are the location of the proximity sensors. Looking at the script in the upper right of the screen shot, it can noted that the last segment, number 29 of 0 - 29, is very close to the target, indicated by the distance of 0.08.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 11, November-2019

Fig. 6 below shows two 10-link arms meeting at the target.

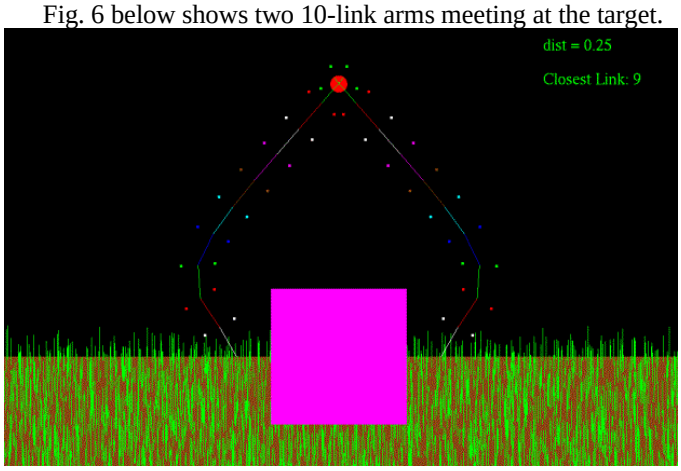

Fig. 6. Two 10-link arms meeting at the target. The target is the red dot. During the simulation, the target is controlled by the mouse.

The simulation is set up so the mouse controls the targets location, leading to the limbs chasing the target about the screen.

IV. RESULTS AND CONCLUSIONS

The simulation results are promising. The arm or arms find their way to the target reliably. Because this technique is iterative, vs. the analytical techniques discussed in the background section, the computation costs are low. While there are multitudes of ways that a 30 joint arm can touch a given target location, the repulsive root technique does not consider them, it only knows if it has reached the target, or if it needs to keep iterating towards it. Because of this, the algorithmic complexity is reduced when compared to a more traditional technique such as the Jacobian Inverse.

Although it was not an expressed goal of the work, the algorithm produces an almost life like movement of the limbs in the simulation. With more polish, this system may be useful in animation work.

Finally, this work is not meant to be the final algorithm of the control system. Instead it is meant to be a flexible exploration algorithm that feeds data to the learning system, which in turn develops the control function/algorithm.

REFERENCES

[1] V. Gupta, R. Chittawadigi, and S. Saha, “RoboAnalyzer: Robot Visualization Software for Robot Technicians,” Proceedings of the Advances in Robotics (AIR '17), Article No. 26, 2017.

[2] K. Erleben, S. Andrews, “Inverse Kinematics Problems with Exact Hessian Matrices,” Proceedings of the Tenth International Conference on Motion in Games (MIG '17), Article No. 14, 2017.

[3] P. Harish, M. Mahmudi, B. Callennec, and R. Boulic, “Parallel Inverse Kinematics for Multithreaded Architectures,” ACM Transactions on Graphics (TOG), 35(2): Article No. 19, 2016.

[4] N. Jaiswal, V. Kumar, “Comparison between Conventional PID and Fuzzy PID Supervisor for 3-DOF Scara type Robot Manipulator,” 2014 IEEE Students' Conference on Electrical, Electronics and Computer Science, DOI: 10.1109/SCEECS.2014.6804431, 2014.

[5] Q. Jiang, and V. Kumar, "The Inverse Kinematics of 3-D Towing," Advances in Robot Kinematics, Ed. Lenarcic, J., Stanisic, M.M., Springer, pp. 321-328, 2010.

[6] I. Chavdarov, R. Trifonov, G. Pavlova, and D. Budakova, “Manipulability and Kinematic Dependences of a Leg of the Six-Legged Robot,” Proceedings of the 19th International Conference on Computer Systems and Technologies (CompSysTech'18), pp. 116 – 119, 2018.

[7] C. Szabo, and Y. Teo, “An Objective-Based Approach for Semantic Validation of Emergence in Component-Based Simulation Models,” Proceedings of the 2012 ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation (PADS '12), pp. 155 – 162, 2012.