IMPROVED EDGE FINDING ALGORITHM: ROBOTIC ARM ROBOT STUDIO

Mitender Kumar[1], Manish Mittal[2], Shobhit Prajapati[3]
[1,2,3]Lecturer, QST, Roorkee, School of Technology, Roorkee

Abstract
Edge finding in Robotics incorporate multiple application includes PICK and PLACE, TRAIN WINDOW FITTING, WEILDING on SHARP EDGES or CAR UNDER BODY etc. Now it became one critical parameter to find the EDGE. The way of finding EDGE is more calculative. The algorithm through which a robot is finding edge must be optimized and short. One quite heavy robot costs on each movement, so minimizing the movements can directly proportional to minimizing the cost. Given algorithm is a slight change in the previous one but used to minimize the edge finding in order of O (n^2).

Keywords: RobotStudio, EDGE finding, COST, ROBOTICS

1. Introduction

Automation is termed as the advancement of modern era. Automation is superset of Robotics and Robotics is not humanoid as per the general convention. There are different
types of Robots. This study enclosed the Industrial Robots, which is specifically called as Industrial Automation.

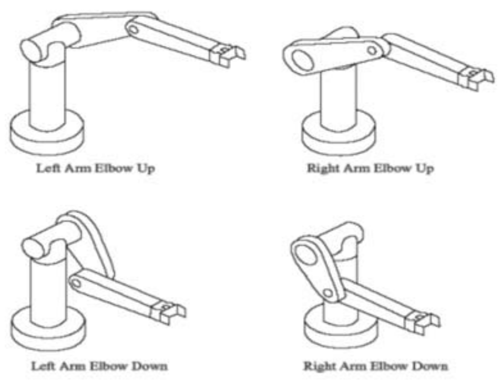

Fig.2: Motions in Robotic Arm

Industrial automation converges highly on ROBOTIC ARM. And one robotic Arm has multiple operations as a whole mentioned below:-

1. *Pick and Place*
2. *Train wall window set*
3. *Welding car under body*
4. *Car Assembling*
5. *Wall Painting*

But all the above mentioned applications need fundamental operation to assemble with. Among all the operations finding edge is very first and critical task, which draws lots of cost in case of electricity and maintenance.

Fig.1: ABB Robotic Arm

1. Ancient Edge finding Algorithm

The classical way to find edge of a sample body was to move robotic arm by n number of steps towards the body. One Proximity Sensor is continuously tracking the presence of edge. The below mentioned algorithm is the correct elaboration of edge detection:

Algorithm: Static Algorithm Procedural Version

```
I.     START
II.    Move 10 MM towards the body.
III.   If Sensor_Out == 0 GOTO step V.
IV.    If Sensor_Out == 1 GOTO step VI.
V.     MOVE 10 MM steps GOTO step III.
VI.    CHANGE the direction.
VII.   MOVE 1 MM towards the body.
VIII.  If Sensor_Out == 0 GOTO step X.
IX.    If Sensor_Out == 1 GOTO step VI.
X.     MOVE 1 MM steps GOTO step VIII.
XI.    CHANGE the direction.
XII.   MOVE 0.1 MM towards the body.
XIII.  If Sensor_Out == 0 GOTO step XV.
XIV.   If Sensor_Out == 1 GOTO step XVI.
XV.    MOVE 0.1 MM towards the body.
XVI.   Record the Coordinates (Say X, Y).
```

Algorithm: Static Algorithm Subroutine Version

Main Routine

```
I.     START
II.    DECLARE signal , distance and
       direction.
III.   INTIALIZE distance by 10 and
       direction by right.
IV.    MovementRoutine(distance,
       dierction).
V.     signal = GetSensorOutcomeRoutine.
VI.    If signal is 0 GOTO  VIII.
VII.   If signal is 1 GOTO  IX.
VIII.  MovementRoutine(distance,dierction
       ),GOTO VI.
IX.    direction = getDirection(direction).
X.     distance = distance / 10.
XI.    MovementRoutine(distance,dierction)
XII.   signal = GetSensorOutcomeRoutine.
XIII.  If signal is 0 GOTO  VIII.
XIV.   If signal is 1 GOTO  IX.
XV.    MovementRoutine(distance,dierction
       ),GOTO VI.
XVI.   direction = getDirection(direction).
XVII.  distance = distance / 10.
XVIII. MovementRoutine(distance,dierction)
XIX.   Record the Coordinates (Say X, Y).
XX.
```

MovementRoutine (distance, direction)

```
I.     MOVE distance in the direction.
```

Get Sensor Outcome Routine

```
I.     If Getting object return 1.
II.    Return 0.
```

GetDirection(Direction)

```
I.     If direction is right change it to left.
II.    If direction is left change it to right.
```

2. New Algorithm- Binary Search

This algorithm performs binary search for the edge. Initial pass goes for **X** distance as per the static parameter. Then it goes for the **X/2** distance with direction change of the movement. And the above mentioned is in the recursive execution. This idea finds EDGE faster than the previous algorithm resulting the less cost also.

Algorithm: Binary Search edge finding

Static Data Structure

1. Speed Of Robot
2. Home Position
3. Object Placement
4. FLAG

Main Routine

```
I.     START
II.    DECLARE
```

moveRoutine(direction)

```
I.     Increment FLAG by 1 on each call
       of routine.
II.    If FLAG reaches to 4 stop the
       routine GOTO VI.
III.   Move in the direction and checks
       for the sensor output signal =
       getSensorOutput.
IV.    direction = getDirection.
V.     moveRoutine(direction)
VI.    Record the Coordinates (Say X,
       Y).
```

getDirection

```
I.     If direction is right change it to
       left.
II.    If direction is left change it to
       right.
```

getSensorOutput

```
I.     If Getting object return 1.
II.    Return 0.
```

3. Comparative Analysis

Comparative analysis of both the algorithm is based on two critical parameters. The parameters are as under:

1. **TIME**
2. **COST**

TIME & COST estimations:

a. By Ancient Algorithm

Geometry of the robotic arm station is given below:
The distance of object from the base of robotic arm is ***X*** (say).
Arm is moving ***10*** units in one move.
Each move stop takes ***t*** time (say).

In 1st Pass

Total stoppage count $C = \frac{X}{10}$

Total Time Taken $T^1 = C * t$

In 2nd Pass

The distance of object from the base of robotic arm is ***Y*** (say).
Arm is moving ***1*** units in one move.
Each move stop takes ***t*** time (say).

Total stoppage count $C = \frac{Y}{10}$

Total Time Taken $T^2 = C * t$

In 3rd Pass

The distance of object from the base of robotic arm is ***Z*** (say).
Arm is moving ***0.1*** units in one move.
Each move stop takes ***t*** time (say).

Total stoppage count $C = \frac{Z}{10}$

Total Time Taken $T^3 = C * t$

Total Time Taken in the Algorithm

$$\boxed{T_{Ancient} = T1 + T^2 + T^3}$$

Suppose machine is consuming cost C per unit of time than Total Cost (TC):

$$\boxed{TC_{Ancient} = T * C}$$

b. Binary Search

In binary search, total numbers of stoppages are 4.

Total time taken

$$\boxed{T_{Binary} = 4 * t}$$

Suppose machine is consuming cost C per unit of time than Total Cost (TC):

$$\boxed{TC_{Binary} = T * C}$$

RESULTS

$$\boxed{T_{Binary} < T_{Ancient}}$$

$$\boxed{TC_{Binary} < TC_{Ancient}}$$

5. Conclusion

We have learnt through our research that there are lots of scope in cost minimization, time reduction and geometry optimization in robotics. Different vendor having different robotic API for his or her own tool. By using those one can minimize the above mention parameter. By using open source library for robotics called ROS (Robot Operating System) any one can start learning robotics and any one can adapt advancement of robotics along with a great career option.

6. References

[1]http://new.abb.com/products/robotics/robotstudio

[2]http://wiki.ros.org/abb/Tutorials/RobotStudio

[3]https://en.wikipedia.org/wiki/Robotic_arm

[4]www.cds.caltech.edu/~murray/books/MLS/pdf/mls94- complete.pdf

[5]. http://www.rapidprogramming.com/

[6]. https://en.wikipedia.org/wiki/RAPID

[7].http://developercenter.robotstudio.com/BlobProxy/manuals/RobotStudioOpManual/doc12.html

[8].http://www.emeraldinsight.com/doi/abs/10.1108/01439910910994605

[9].https://www.researchgate.net/publication/235253521_Technology_and_applications_of_ABB_RobotStudio

[10].http://dl.acm.org/citation.cfm?id=1231103