Cheap and Fast Implementation of Linear and Cubic Piecewise Interpolation for Robot Path Smoothing on Arduino Uno Board

Trias Prima Satya[1], Jans Hendry[1*], Aditya Putra Yudhananta[1], Zubainindra Bagus F Meliawan[1]
[1]Department of Electrical Engineering and Informatics, Universitas Gadjah Mada; trias.primasatya@ugm.ac.id, adityapy28@mail.ugm.ac.id, zubainindra_agus@mail.ugm.ac.id
[*]Correspondence: jans.hendry@ugm.ac.id

Intisari – Teknologi* mobile robot *telah menjadi bagian dari keseharian manusia. Perkembangannya yang cepat telah menghasilkan robot–robot yang digunakan untuk tujuan khusus seperti layanan kebersihan, memindahkan barang di gudang, dan otomasi lainnya. Robot–robot ini biasanya memiliki kemampuan navigasi otomatis yang dihasilkan dari pelatihan berupa peta jalur yang akan dilalui. Permasalahannya adalah cukup banyak metode dengan tujuan tersebut yang tidak menyertakan metode penghalusan jalur saat membelok. Penambahan teknik tersebut memang memberatkan kerja otak robot apalagi jika metode yang digunakan cukup kompleks sehingga membutuhkan perangkat keras yang mahal. Penelitian ini mengusulkan pendekatan metode* linear *dan* cubic *terhadap persamaan aslinya. Hasilnya menunjukkan bahwa nilai* mean squared error *keduanya 0,7789 dan 0,7365 dengan waktu komputasi sekitar 0,809 detik dan 0,836 detik. Maka kedua pendekatan ini cukup menjanjikan untuk diterapkan pada perangkat keras yang murah untuk aplikasi* mobile robot*.

***Kata kunci* – mobile robot*, penghalusan jalur, interpolasi,* linear, cubic**

Abstract – Mobile robot technology has become our daily need. The pace of development has yielded various useful robots that can do specific tasks like cleaning services, transporting stuff in a warehouse, and automation. These robots usually have self-navigation ability that comes from a pre-training map in the form of trajectories. The problem is most of the methods do not consider the path smoothing at a turning point. This is because it can burden the robot processor. Hence, in this research, we implemented an estimation of the two most popular methods in path smoothing based on interpolation which are linear and cubic interpolations. The result shows that the mean squared error between the estimated and original formula of interpolation is around 0.7789 and 0.7365 with execution times around 0.809 seconds and 0.836 seconds, respectively. Hence, both estimation methods can be very promising to be implemented in real-world mobile robots application.

Keywords – mobile robot, path smoothing, interpolation, linear, cubic

I. INTRODUCTION

Mobile robots have become a familiar technology nowadays. They are used for specific purposes with specific tasks like cleaning, transporting stuff in a warehouse, and industrial automation [1]. These robots are developed with the ability to self-navigating and survive in an uncommon environment. For this purpose, a robot should be equipped with various sensors, for instance, inertial measurement units (IMU), RBG and depth camera, range finders (laser-based or ultrasonic), and so on. In many cases, these robots are required to create a full path that can be a closed loop or an open loop. This path is regarded as their map to use for future moves when they are burdened with the same tasks. One of the popular methods for self-navigating mobile robots is SLAM (simultaneous localization and mapping) method which helps them to build path or trajectory maps and localize their position in that map [2]. Path creation in this discussion is called path planning.

Quite many of the algorithms used for path planning end up with schools of straight lines and sharp turns to avoid collision with static obstacles as shown in Figure 1. Commonly, path planning follows *straight blue lines* with sharp turns that can be disadvantageous. When a robot finds a turn, it will be slowing down and stop then accelerate in another direction. This action can be expensive in terms of battery source as overshoot might happen each time it accelerates. From another point of view, suppose that the robot is a service robot chair that transports a disabled person, it has the potential to cause shock and inconvenience to the person. Added to that, the kinematics of robots might not be able to handle such turns. Hence, it is safer to create a smooth path at the turning point as depicted by the *orange line*.

To create a smooth path as shown by the *orange line*, some algorithms can be used. They are polynomial interpolation, Bezier curve, Cubic Splines, B-Spline, NURBS curve, Dubin's curve, Clothoid, Hypocyloid, and other optimization-based curves [3]. Simply, this turning point depicted by *the blue lines* is called discontinuity. This point should be replaced with a new line that can bridge two consecutive lines which is called a curve. To create a smooth and better curve, an equation with high complexity is needed. Hence the implementation in terms of execution time and resource availability can be hard. So, an estimation is needed with an accepted error tolerance as such they can be planted into cheap hardware with fast execution time.

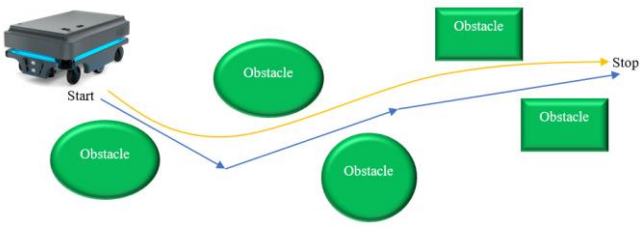


Figure 1. Illustration of mobile robot path

In this paper, the implementation of two famous path smoothing methods estimation is introduced. They are linear and cubic piecewise polynomial interpolations. The implementation is applied in Arduino Uno which has lower features than others. To have better reading experiences, this paper is organized as follows: 1) the motivations and aims are explained in the Introduction, 2) the methods used in this research are explained in the Methodology, 3) the results and discussions are described in the Result and Discussion, 4) the Conclusion.

II. METHODOLOGY

The interpolation technique is frequently used in many fields like signal processing, image processing, data analysis, economics, electronics, controls, and so on. This technique is first introduced by Waring [4]. In a nutshell, for any pairs of (x_i, y_i), new members of y_i at new points x_i that lies between them can be estimated by involving the given y_i.

A. Linear Interpolation

Assume that *f(x)* is a quadratic formula that represents a smooth curve for the robots' path. The linear interpolation is used to fill the in-between points as such the robots can move smoothly without further slowing down.

$$f(x) = ax^2 + bx + c \tag{1}$$

the first derivative of *f(x)* is calculated by (2).

$$f'(x) = 2ax + b \tag{2}$$

with the range of x = [0, 1], then the value of *f(x)* and its derivative are

$$f(0) = c \tag{3}$$

$$f(1) = a + b + c \tag{4}$$

$$f'(0) = b \tag{5}$$

$$f'(1) = 2a + b \tag{6}$$

by assuming that p_0 and p_1 are points at $x = 0$ and $x = 1$, respectively, then the value of *a*, *b*, and *c* can be replaced with

$$a = \frac{p_1 - p_0}{2} \tag{7}$$

$$b = \frac{p_1 - p_0}{2} \tag{8}$$

$$c = p_0 \tag{9}$$

Hence, the quadratic curve (*s(x)*) can be fed to the microcontroller directly as expressed by (10).

$$s(x) = \frac{p_1 - p_0}{2}x^2 + \frac{p_1 - p_0}{2}x + p_0 \tag{10}$$

B. Cubic Interpolation

In this technique, the curve is assumed to be a 3rd-order polynomial as stated in (11). The goal is like (10), to find the new assignment of *a*, *b*, and *c* that can be directly filled with numerical values which is the position of robots.

$$f(x) = ax^3 + bx^2 + cx + d \tag{11}$$

The derivative of *f(x)* is

$$f'(x) = 3ax^2 + 2bx + c \tag{12}$$

with the range of x = [0, 1], then the value of *f(x)* and its derivative are

$$f(0) = d \tag{13}$$

$$f(1) = a + b + c + d \tag{14}$$

$$f'(0) = c \tag{15}$$

$$f'(1) = 3a + 2b + c \tag{16}$$

this technique takes four points (two on the left and two on the right) to calculate the final estimation. Assume that p_0, p_1, p_2, and p_3 are values at $x = -1$, $x = 0$, $x = 1$, and $x = 2$, respectively. Then the value of *a*, *b*, *c*, and *d* are

$$a = -\frac{1}{2}p_0 + \frac{3}{2}p_1 - \frac{3}{2}p_2 + \frac{1}{2}p_3 \tag{17}$$

$$b = p_0 - \frac{5}{2}p_1 + 2p_2 - \frac{1}{2}p_3 \tag{18}$$

$$c = -\frac{1}{2}p_0 + \frac{1}{2}p_2 \tag{19}$$

$$d = p_1 \tag{20}$$

by substituting (17) – (20) to *f(x)*, the final *s(x)* can be calculated as in (21).

$$s(x) = \left(-\frac{1}{2}p_0 + \frac{3}{2}p_1 - \frac{3}{2}p_2 + \frac{1}{2}p_3\right)x^3 + \left(p_0 - \frac{5}{2}p_1 + 2p_2 - \frac{1}{2}p_3\right)x^2 + \left(-\frac{1}{2}p_0 + \frac{1}{2}p_2\right)x + p_1 \tag{21}$$

C. Mean Squared Error

Let $s_n[x]$ and $\hat{s}_n[x]$ denote the original equation and estimation of the polynomial interpolation, hence the error can be calculated using mean squared error using (22).

$$mse = \frac{1}{N}\sum_{n=1}^{N}(s_n[x] - \hat{s}_n[x])^2 \tag{22}$$

III. RESULT AND DISCUSSION

The microcontroller used in this experiment is ATMEGA328P which is used in the Arduino Uno development board as shown in Figure 2. Its specification is shown in Table 1. With these low features, it is better to implement an interpolation method with low complexity while maintaining quality. Hence, the approach offered in this experiment can meet the requirement.

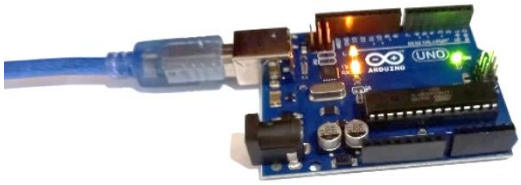


Figure 2. Arduino Uno development board

Table 1. ATMEGA328P specifications

Type	Value
CPU	8-bit AVR
Speed	1 MIPS for 1 MHz
RAM	2 Kbytes SRAM
EEPROM	1 Kbytes
Type	Value

A. Linear Interpolation

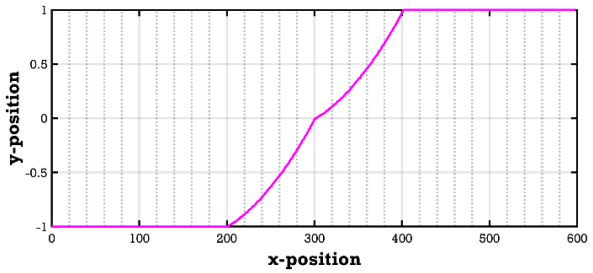
The interpolation of consecutive points that represent robot position (*x*, *y*) using the estimation method is shown in Figure 3(a). These values are read by MATLAB via serial communication (*port com:6*) right after the calculation is finished. Curve plots using *serial plot* which is available in Arduino IDE can't provide high-definition figures for better visual. Hence, we make another program to read and display the curve in another programming language. In figure 3(b), the curve calculated using original interpolation with the same (*x*, *y*) is plotted using MATLAB as a comparison. As can be observed that they have similar tendencies, but the estimated curve has exhibited few errors. There is a discontinuity that happens on (*x*, *y*) = (300, 0) which is caused by the *sign* changes in the *y*-direction. It will not happen in the real-world coordinates as their value will be always positive.

B. Cubic Interpolation

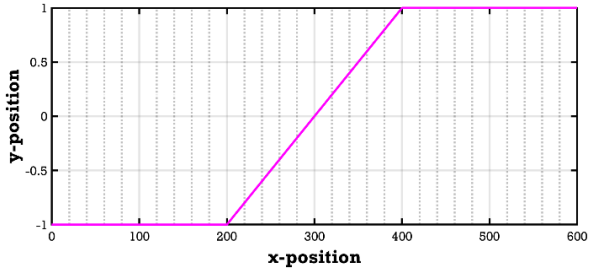
The cubic interpolation over the mobile robot's path is shown in Figure 4. Like linear interpolation, the cubic estimation also produces a similar curve compared with the original cubic interpolation but with some errors depicting discrepancies between the estimated and original cubic. The curve from the estimated cubic has 2 peaks because they are rolling down faster than the original. That is why they yielded errors between interpolated points.

C. Mean Squared Error

The error between estimated and original interpolation is also measured in this experiment. It can be shown that the interpolation error for estimated linear and cubic interpolation is around 0.7789 and 0.7365, respectively. The smaller error that cubic has indicated that this estimation is better in performance compared to estimate linear. Hence, the estimated cubic approach that we used can be potentially useful in a real-world implementation.

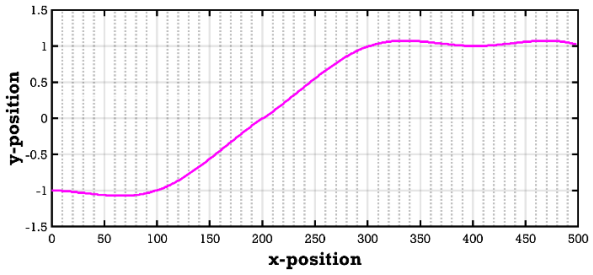


(a)

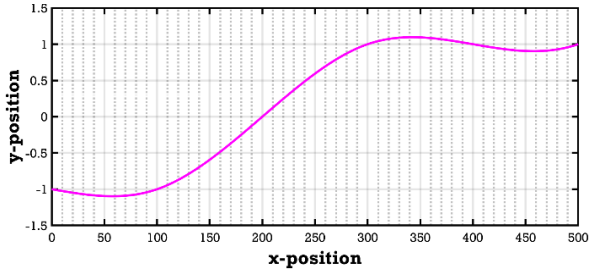


(b)

Figure 3. Path smoothing by (a) estimated linear interpolation, (b) original linear interpolation



(a)



(b)

Figure 4. Path smoothing by (a) estimated cubic interpolation, (b) original cubic interpolation

D. Execution Time

The execution time is measured by using *millis()* which is a built-in function in Arduino IDE. When measured, the linear interpolation takes around 0.809 seconds to finish one estimation and 0.836 seconds for cubic interpolation. The time difference between both methods is slightly low but cubic interpolation can give a smooth curve than linear interpolation with lower errors. Both methods executed the iteration in less than one second which is very promising.

IV. CONCLUSION

The polynomial interpolations for the mobile robot's path smoothing have been implemented in the Arduino Uno board. The execution time for linear and cubic interpolation is around 0.809 sec and 0.836 sec, consecutively. They are still under 1 second which is probably acceptable for a mobile robot to plan its smooth path. The curve yielded from the estimations is similar to the original equation of the polynomial. Their MSE are 0.7789 and 0.7365, respectively, which indicates that the estimated formula can be very promising to be used in a real-world implementation.

ACKNOWLEDGEMENT

This is part of the main research sponsored by the Department of Electrical Engineering and Informatics and Vocational College of Universitas Gadjah Mada under a *competitive scheme*.

REFERENCES

[1] N. Correll *et al.*, "Analysis and observations from the first amazon picking challenge," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 1, pp. 172–188, 2016.

[2] A. A. Ravankar, Y. Hoshino, A. Ravankar, L. Jixin, T. Emaru, and Y. Kobayashi, "Algorithms and a framework for indoor robot mapping in a noisy environment using clustering in spatial and hough domains," *Int J Adv Robot Syst*, vol. 12, no. 3, p. 27, 2015.

[3] A. Ravankar, A. A. Ravankar, Y. Kobayashi, Y. Hoshino, and C.-C. Peng, "Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges," *Sensors*, vol. 18, no. 9, p. 3170, 2018.

[4] E. Waring, "Vii. problems concerning interpolations," *Philos Trans R Soc Lond*, no. 69, pp. 59–67, 1779.