Improved DDA Algorithm and FPGA Implementation

Yi-Xuan WANGª, Quan-Gang WENª, He YINª, Zhi-Gang HAO, Xin-Zheng DONG

Lab building 430, Computer Department, Zhuhai College of Jilin University, Zhuhai, Guangdong, China
ªyixuan_jessica@hotmail.com, ªwen_sir_125@163.com, ª34089689@qq.com

Keyword: Interpolation, DDA algorithm, FPGA, Errors analysis.

Abstract. In this paper, the basic principle of DDA interpolation algorithm is discussed and an improved algorithm is designed based on errors analysis of DDA algorithm. These two algorithms are simulated on Matlab, testing result shows chord error δ of improved DDA algorithm was reduced by 20% - 50% on interpolation accuracy. Finally the algorithm is implemented on FPGA to test its real-time performance and interpolation speed.

Introduction

Interpolation was widely accepted in traditional machinery manufacturing decades ago. In engineering and science, one often has a number of data points, obtained by sampling or experimentation, which represent the values of a function for a limited number of values of the independent variable. It is often required to interpolate (i.e. estimate) the value of that function for an intermediate value of the independent variable. This may be achieved by curve fitting or regression analysis [1]. With the popularity of industrial automation and intelligent control, motion control system has become the core technology of universal Industry 4.0, and interpolation algorithm is essential to modern motion control system.

Many traditional interpolation algorithms have been proposed by universal researchers, such as point by point comparison method, digital differential analyzer (DDA algorithm), and vector discriminant method. DDA algorithm is an interpolation generation method in computer graphics based on linear differential equation. Not as widely used as point by point comparison method, DDA algorithm has floating calculation and rounding operation which are not easily implemented on hardware.

The rest of this paper is organized as follows; basic principle of DDA algorithm is analyzed in the second section; next section proposes some improvements based on error analysis of DDA algorithm; the forth section describes simulation of the algorithm on Matlab and implementation on FPGA. The fifth section is the conclusion.

Figure 1. Integral Operation of Function y = f(x).
The Basic Principle of DDA Algorithm

Set a function \( y = f(x) \), as shown in Fig.1, from the geometry expression, integral operation is the size of the area surrounded by the function curve \( S \), as shown in equation (Eq.1).

\[
S = \int_0^t y \, dt \\
S = \int_0^t y \, dt = \sum_{i=0}^{n} y_i \Delta t \\
S = \sum_{i=0}^{n} y_i
\]

The size of the area \( S \) can be approximate to the sum of many small rectangular areas; the width of rectangles is an independent variable \( \Delta t \), and the height is the corresponding \( f(x) \) on Y-axis (shown as Eq.2). This is so called the rectangular integral method, and the formula is called the rectangle formula. In practical, if we set \( \Delta t = 1 \), one pulse quantity, The formula Eq2 can be simplified to Eq3. It shows that integral operation can be transformed into summation operation, if variable \( \Delta t \) is small enough, so as the pulse is small enough, the error of using add operation instead of integral can be restricted within the range of allowable.

So we can use a function register \( J_V \), an accumulator register \( J_R \) and an adder \( Q \) to complete integral operation. Once a pulse signal came, we accumulate the value of \( J_V(y_i \text{ at current}) \) into accumulator register \( J_R \). If we take the capacity of register \( J_R \) as an area unit, every overflow pulse of register \( J_R \) in the process of accumulating indicate an area unit obtainment. So the total amount of overflow is as integral area of \( S \).

DDA Linear Interpolation

Now we extend the former principle to linear interpolation. Set the processing line is OZ, coordinates of start point O is (0, 0) and coordinates of end point Z is \((x_z, y_z)\), shown as Fig. 2. Set length of OZ is \( L \), rate of feeding is \( v \) and we obtain x and y coordinates at current time \( t \) is:

\[
x = \frac{v}{L} x_z t, \quad y = \frac{v}{L} y_z t
\]

Differential equation is:

\[
\frac{dx}{dt} = \frac{v}{L} x_z \quad \text{or} \quad dx = \frac{v}{L} x_z \, dt
\]
\[
\frac{dy}{dt} = \frac{v}{L} y_z \quad \text{or} \quad dy = \frac{v}{L} y_z \, dt
\]
Take v/L as federate K, dx, dy and dt in incremental form \( \Delta x \), \( \Delta y \) and \( \Delta t \):

\[
\Delta x = K x_z \Delta t \quad \Delta y = K y_z \Delta t
\]  

(6)

Thus, we can use two function register \( J_{RX} \), \( J_{RY} \) to preserve the value of \( x_z \) and \( y_z \) and two accumulator registers \( J_{RX}, J_{RY} \) to do the linear interpolation. Take two-dimensional linear interpolation as an example, straight line start at point (0, 0) end at point (7, 5). According to the coordinate difference, take 3 bit register as the accumulator, we got the operation form (Table 1) and plot graph (Fig.3).

Assuming that forward toward the end point \( v \) (distance) for each pulse \( \Delta t \), in other words, uniform motion to the finish point at the speed of \( v \), thus we obtain the following formula:

\[
\frac{v}{oz} = \frac{v_x}{x_z} = \frac{v_y}{y_z} = K
\]  

(7)

K is so-called scaling factor. Set the number of step from start to finish is m, we have Eq.8.

\[
m*K = 1, \quad \text{so as} \quad K = \frac{1}{m}
\]  

(8)

When choosing m and K, m must an integer, we must consider that each increment of \( \Delta x \) and \( \Delta y \) must be no more than 1. Maximum value of \( x_z \) and \( y_z \) are limited by register capacity. Assume that n-digit register, the maximum value of \( x_z \) and \( y_z \) is \( 2^n-1 \). To meet the conditions of \( \Delta x \) and \( \Delta y \) must be no more than 1, the following formula Eq6 can be deduced.

\[
\Delta x = Kx_z = K(2^n-1) < 1
\]  

(9_1)

\[
\Delta y = Ky_z = K(2^n-1) < 1
\]  

(9_2)

\[
K < \frac{1}{2^n-1} \text{ In general set } K = \frac{1}{2^n}
\]

| Calculate counter | \( J_{RX} \) | \( J_{RY} \) | \( \Delta x \) | \( \Delta y \) | \((x_i, y_i)\) |
|------------------|-------------|-------------|-------------|-------------|----------------|
| 0(Init)          | 0           | 0           |             |             | (0,0)          |
| 1                | 7           | 5           | 0           | 0           | (1,1)          |
| 2                | 14=8+6      | 10=8+2     | 1           | 1           | (2,1)          |
| 3                | 13=8+5      | 7           | 1           | 0           | (3,2)          |
| 4                | 12=8+4      | 12=8+4     | 1           | 1           | (4,3)          |
| 5                | 11=8+3      | 9=8+1      | 1           | 1           | (5,3)          |
| 6                | 10=8+2      | 6           | 1           | 0           | (6,4)          |
| 7                | 9=8+1       | 11=8+3     | 1           | 1           | (7,5)          |
| 8                | 8           | 8           | 1           | 1           |                |
DDA Circular Interpolation

To generate arc with DDA algorithm works as follows, assuming that arc AB is located in the first quadrant, counterclockwise direction, start at point A\((x_a, y_a)\), end at point B\((x_b, y_b)\) and radius is R. Feeding speed along the arc tangent is constant \(v\), which can be resolve into horizontal velocity \(v_x\) and vertical velocity \(v_y\) at any point \(N(x_n, y_n)\) on arc. The following formula can be derived:

\[
\begin{align*}
v_x &= \frac{dx_n}{dt} = -v \sin \theta = -\frac{v}{R} y_n = -\left(\frac{v}{R}\right)y_n \\
v_y &= \frac{dy_n}{dt} = v \cos \theta = \frac{v}{R} x_n = \left(\frac{v}{R}\right)x_n
\end{align*}
\]

(10_1)

(10_2)

Transform into the following Eq.11.

\[
\begin{align*}
dx_n &= -\left(\frac{v}{R}\right) y_n dt \text{ or } \Delta x_n = -\left(\frac{v}{R}\right) y_n \Delta t \\
dy_n &= \left(\frac{v}{R}\right) x_n dt \text{ or } \Delta y_n = \left(\frac{v}{R}\right) x_n \Delta t
\end{align*}
\]

(11_1)

(11_1)

Take \(v/R\) as federate \(K\), \(dx\) \(dy\) and \(dt\) in incremental form \(\Delta x\) \(\Delta y\) and \(\Delta t\):

\[
\begin{align*}
\Delta x_n &= -K y_n \Delta t \\
\Delta y_n &= K x_n \Delta t
\end{align*}
\]

(12)

Similar as linear interpolation, we can use two function register \(J_{X}(\text{for value of } y_n)\)
\(J_{Y}(\text{for value of } x_n)\) to preserve the value of \(y_n\) and \(x_n\) and two accumulator registers \(J_{RX}, J_{RY}\) to do the linear interpolation.

Errors Analysis and Improvements of DDA Interpolation

The basic cause of the error of DDA interpolation is step-by-step processing; each inaccurate calculation is producing errors on every step. There are several methods to help us to reduce these errors.

Reduce the Pulse Equivalent

As the pulse equivalent reduced, the error’s geometry size decrease and the number of steps increase synchronously. For instance, line OZ, Fig. 5 display us the difference between
step-size is 1(Left) and step-size is 0.2(Right). Decreasing of pulse equivalent has to increase calculation speed, which is limited by hardware implementation, to obtain the same motion speed. As previously discussed, this method is practical limited by hardware minimum pulse equivalent and calculation speed.

![Graph: Reduce the Pulse Equivalent](image1.png)

a) step-size=1  b) step-size=0.2

Figure 5. Reduce the Pulse Equivalent.

**Preset Accumulator to 0.5 Capacity**

The accumulator will overflow only to $2^n$ in the original DDA algorithm. We can obtain error reduction if we preset accumulator to half of its capacity instead of 0 before the accumulation step. The basic idea of preset is just like rounded integer is practically more accurate than floored integer. Take linear interpolation as an example, line $OZ (Z_x=8, Z_y=2)$. Interpolation trajectory of original DDA algorithm and preset accumulator shows in figure 6, and latter error significantly reduced. Similar error reduction has been achieved in circular interpolation.

![Graph: Preset Accumulator to 0.5 Capacity](image2.png)

a) Original DDA algorithm  b) preset accumulator DDA algorithm

Figure 6. Preset Accumulator to 0.5 Capacity.

**Use Secant Instead of Tangent on Circular Interpolation**

In circular interpolation, using secant instead of tangent could achieve more accurate approximation accordingly improve the interpolation precision. Fig. 7 shows the principle of improved algorithm; Fig. 8 shows first quadrant circular interpolation using tangent approximation and secant approximation.
In processing the first quadrant circular arc, radius is R. Current point is $A_{i-1}$, next point in original DDA is $A_i$. Clearly we can see, deviation of point $A_i$ from circular trajectory is far, the radial error is very large. We could apply the following step to correct direction of feed.

1. Calculate $A_i$ from original DDA algorithm.
2. Calculate coordinates of midpoint B of line $A_{i-1}A_i$.
3. Calculate length $L_1$ from center O to point B.
4. Take O as center, $L_1$ as radius, calculate the unit direction vector at point B on circular arc, which is the BC direction.
5. Calculation unit direction vector of point B in X and Y axes direction ($\Delta X_i, \Delta Y_i$)
6. Next point $A_i$ can be calculate by adding $DX_i, DY_i$ from $A_{i-1}$ ($X_i = X_{i-1} + \Delta X_i$, $Y_i = Y_{i-1} + \Delta Y_i$).

Interpolation precision can be improved by 30-40 percent by applying the above improvement in Matlab simulation. But the computational complexity of this method increase 2 times, which might slow down the interpolation speed.

**Simulation and FPGA Implementation of Improved DDA Interpolation**

The original DDA algorithm and its improved strategy are simulated in Matlab environment. The simulation results shown in Figure 5, Figure 6 and Figure 8 give us basic concept of accuracy improvements of these strategy. We introduce the chord error $\delta$ to measure the accuracy of interpolation algorithms in our test. Chord error $\delta$ was reduced by 20% - 50% according to different test cases and related parameters, but we achieve very little increase of interpolation speed.

Accuracy is the most important value of interpolation algorithm, and interpolation algorithm is the core value of motion control. Yet its stability and real-time performance practical is crucial after all. To test its real-time performance, our team has implemented on FPGA. Its characteristics of small size, high integration, short development cycle makes it suitable test environment for high speed motion control. The simulation results are shown in Fig. 9, upper one is linear interpolation test case, bottom one is circular interpolation test case, and program is realized in verilog language.
Conclusion

Based on errors analysis of original DDA algorithm, the improved algorithm proposed in this paper has improved in local accuracy and global accuracy obviously. The improved algorithm is simulated in Matlab and realized in FPGA chip, its real-time performance and stability has been validated in practice.

References

[1] https://en.wikipedia.org/wiki/Interpolation.

[2] Zhang Bingxing. Error analysis of DDA circular interpolation and solution [J]. Value Engineering, 2013-02.

[3] Li En Lin. Principle of Interpolation [M], Peking: China Machine Press, 1984: 35-39.

[4] Zhao Wei. Research on Interpolation Algorithm and Acceleration-Deceleration Controller in CNC System [D]. Tian Jin: Tian Jin University, 2004.

[5] Li Fen, Wang Hai-tao. Design of arc interpolation program based on extended DDA[J], Manufacturing Automation, 2012, 34(144-146).

[6] Tang, Huifeng. Research and Simulation of Improving Point by Point Comparison Method Circular Arcs Interpolation [J], Microcomputer Information, 2006, 22(5-3).