International Journal of Computer Engineering Research Vol. 2(3), pp. 51-59, April 2011
Available online at http://www.academicjournals.org/IJCER
ISSN 2141- 6494 ©2011 Academic Journals

Full Length Research Paper

A shift-add algorithm for computing Bezier curves

GU Feng

ZheJiang Technical Institute of Economics, Hangzhou, China. E-mail:coolfun630@hotmail.com

Accepted 29 March, 2011.

A shift-add algorithm based on coordinate rotation digital computer algorithm for computing Bezier curves was presented in this paper. This algorithm can be implemented in basic computing system (which deals only with shift, add and logical operations) which exists in many areas. Convergence of the algorithm was proved. Error estimation was analyzed. A numerical experiment was carried out to validate algorithm's effectiveness and efficiency.

Key words: Bezier curve, shift-add algorithm, basic computing system, CORDIC, approximation.

INTRODUCTION

Bezier curves are very popular in CAD/CAM and other curve fitting systems. Bezier spline of degree *n* (order *n*+1) can be obtained by deCasteljau algorithm as an interpolation between (*n*+1) control points P_0, P_1, ... , P_n. The general expression for Bezier curve $P(t)$ of degree *n* (order *n*+1) is

$$B_0^0(t) = 1,$$

$$B_i^k(t) = (1-t)B_i^{k-1}(t) + tB_{i-1}^{k-1}(t), \ (k=1, 2, , n.\ i=0, 1, \ldots, k.)$$

$$P(t) = \sum_{i=0}^{n} B_i^n(t) P_i$$

where $B_i^k(t)$ (k=0,1,2,…,n. i=0,1,…,k) are Bernstein polynomials, P_i (i=0,1,…,n) are control points (Aumann, 1997; Catherine and Sonya, 2003; Farin 1997).

Different methods for constructing Bezier curves were developed (Catherine and Sonya, 2003; Han et al., 2008; Popiel and Noakes, 2006). Existing algorithms for constructing Bezier curves usually run in advanced computing systems. In this paper we discuss how to computing Bezier curves in a basic computing system. A basic computing system deals only with shift, add and logical operations. Basic computing systems exist in many application systems such as industrial control systems, military application systems, medical application systems, etc. Single chip microcomputers and Field Programmable Gate Arrays (FPGA) are good examples. Coordinate rotation digital computer (CORDIC) algorithms are well known shift-add algorithms for computing a wide range of elementary functions including trigonometric, hyperbolic, linear and logarithmic functions (Volder, 1959; Muller, 2006; Neil, 1998). CORDIC algorithms have been generalized (Feng, 2006; Xiaobo et al., 1991). Their convergences and error estimations have been analyzed (Feng, 2006). With the development of hardware technique, these fast-united shift-add algorithms can be implemented in hardware system (even multipliers need not be used) (Ray, 1998). These algorithms can also be coded with assembly language.

In this paper a shift-add algorithm based on CORDIC algorithm for computing Bezier curves is presented.

Variables that are made used of include:

A, $(a_{ij})_{n\times n}$: matrixes.

$B_i^k(t)$ (k=0,1,2,…,n. i=0,1,…,k): Bernstein polynomials of degree k.

BSx, BSy, bst: coordinates of a point on a Bezier curve.

$b_j^i(t)$: real values of Bernstein polynomials of degree i.

$bs(t)$: real value of a Bezier curve.

$F_N(x,\delta)$: x's measurement expansion with $\{\delta_i\}_0^\infty$.

I: identity matrix.

i,j,k,n,m,N: integer variables.

P_i (i=0,1,…,n): control points.

P_i^x, P_i^y, P_i^u (i=0,1,…,n): coordinates of a control point P_i.

$R(\delta)$: the measurement radius of a normal series

$\{\delta_i\}_0^\infty$.

$sg(x)$: the sign function.

s, s_i (*i*=0,1,…): sign of a number.

t,u,v,w, w_1 ,*x,y,z,* x_i , y_i , z_i (*i*=0,1,…): real variables.

ε , ε_i (*i*=0,1,…): the upper bound of error.

$\{\delta_i\}_0^\infty$: a normal series.

DESCRIPTION OF THE ALGORITHM

A sign function and a positive number series (Feng, 2006) are defined as follows;

$$sg(x)=\begin{cases}1 & x\geq 0\\ -1 & x<0\end{cases}$$

$$\{\delta_i\}_0^\infty = \{2^{-i}\}_0^\infty$$

Let $\{P_i=(P_i^x,P_i^y)\}_0^n$ be control points. $B_i^k(t)$ (*k*=0,1,2,…,*n*. *i*=0,1,…,*k*. $t\in[0,1]$) denote Bernstein polynomials,. ε is the upper bound of error. The shift-add algorithm for computing Bezier curves consists of one main program and three subprograms.

Subprogram 1: UV(u,v,ε).

1° if *u=0* or *v=0* then result:=0. Stop.

2° *s*:=1.

if *u<0* then

begin

s:=-*s*;

u:=-*u*.

end;

if *v<0* then

begin

s:=-*s*;

v:=-*v*.

end;

3° *m*:=0;

while $u>1$ do

begin

$u:=\frac{u}{2}$;

m:= *m*+1.

end;

4° *N*:=2;

$\varepsilon:=2^{-m}\times\varepsilon$;

while $\delta_{N-2}>\varepsilon$ do *N*:=*N*+1;

5° *i*:=1; $x_1:=u$, $y_1:=v$, $z_1:=0$.

6° while *i*<*N* do

begin

$s_i:=sg(x_i)$;

$x_{i+1}:=x_i+s_i\times\delta_i$;

$z_{i+1}:=z_i+s_i\times\delta_i\times y_1$;

i:=*i*+1;

end;

7° $z_N:=s\times 2^m\times z_N$.

result: = z_N . Stop.

Subprogram2: Bernstein (n,t,ε).

1° $B_0^0:=1$;

for *i*:=1 to n

begin

$B_{-1}^i:=0$;

$B_{i+1}^i:=0$;

end;

2° $\varepsilon_1:=\varepsilon\times 2^{-1}$;

while $UV(\varepsilon_1,n,\varepsilon\times 2^{-1})>\varepsilon$ do $\varepsilon_1:=\varepsilon_1\times 2^{-1}$;

3° for *i*:=1 to n

for *j*:=0 to *i*

$B_j^i(t):=$ UV($1-t,B_j^{i-1}(t),\varepsilon_1$)+UV($t,B_{j-1}^{i-1}(t),\varepsilon_1$);

4° Output $B_j^n(t),\ j=0,...,n.$ Stop.

Subprogram3: BS($n,t,P_{i=0\ldots n}^u,\varepsilon$).

1° *w*:=0;

for *i*:=0 to *n*

$w:=w+sg(P_i^u)\times P_i^u$;

w:=*w*+1;

$w_1:=UV(w,n+1,\varepsilon\times 2^{-1})$;

$\varepsilon_1:=\varepsilon$;

while $UV(\varepsilon_1,w,\varepsilon\times 2^{-1})>\varepsilon$ do $\varepsilon_1:=\varepsilon_1\times 2^{-1}$;

$\varepsilon_2:=\varepsilon$;

while $UV(\varepsilon_2,w_1,\varepsilon\times 2^{-1})>\varepsilon$ do $\varepsilon_2:=\varepsilon_2\times 2^{-1}$;

2° Bernstein (n,t,ε_1);

bst:=0;

3° for *i*:=0 to *n*

$bst:=bst+UV(P_i^u,B_i^n(t),\varepsilon_2)$;

4° Output *bst.* Stop.

Main program: Bezier Curve $(n,t,P_{i=0...n},\varepsilon)$

1° BSx =BS($n,t,P^x_{i=0...n},\varepsilon$).

2° BSy =BS($n,t,P^y_{i=0...n},\varepsilon$).

3° Output BSx , BSy . Stop.

Remarks,

1. Only operations shifting (i.e. $2^{-i}\times t$) and adding were concerned in the algorithm.
2. The determination of iteration number N in Subprogram 1 is based on Theorem 1.
3. In Subprogram 2, $\varepsilon_1 \le \dfrac{\varepsilon}{2n}$.
4. In Subprogram 3, $\varepsilon_1 \le \dfrac{\varepsilon}{\sum_{i=0}^{n}\left|P_i^u\right|+1}$,

$$\varepsilon_2 \le \frac{\varepsilon}{(n+1)\left(\sum_{i=0}^{n}\left|P_i^u\right|+1\right)}.$$

5. Experience showed that Subprogram 1 usually runs not more than 28 steps, so it is a fast algorithm.

CONVERGENCE AND ERROR ESTIMATION OF THE ALGORITHM

$\{\delta_i\}_0^{\infty} = \{2^{-i}\}_0^{\infty}$ is a normal series, with measurement radius $R(\delta)=\sum_{i=0}^{\infty}\delta_i = 2$ (Feng, 2006).

Let $A=\begin{pmatrix}0 & 0\\ 1 & 0\end{pmatrix}$. There are $A^i=\begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix}(i=2,3,.....)$,

and $e^{Ax}=\sum_{i=0}^{\infty}\frac{(Ax)^i}{i!}=I+Ax=\begin{pmatrix}1 & 0\\ x & 1\end{pmatrix}$.

Let $x\approx F_N(x,\delta)=\sum_{i=0}^{N}sg(x_i)\delta_i$ be *x*'s measurement expansion with $\{\delta_i\}_0^{\infty}$ $R(\delta)=\sum_{i=0}^{\infty}\delta_i=2$ (Feng, 2006).

There is,

$$\begin{pmatrix}1 & 0\\ x & 1\end{pmatrix}=e^{Ax}\approx e^{A\sum_{i=0}^{N}sg(x_i)\delta_i}=\prod_{i=0}^{N}e^{Asg(x_i)\delta_i}=\prod_{i=0}^{N}\begin{pmatrix}1 & 0\\ sg(x_i)\delta_i & 1\end{pmatrix},$$

$$\begin{pmatrix}y\\ xy\end{pmatrix}=\begin{pmatrix}1 & 0\\ x & 1\end{pmatrix}\begin{pmatrix}y\\ 0\end{pmatrix}\approx\prod_{i=0}^{N}\begin{pmatrix}1 & 0\\ sg(x_i)\cdot\delta_i & 1\end{pmatrix}\begin{pmatrix}y\\ 0\end{pmatrix}.$$

It is equivalent to the following iterative process,

$$\begin{cases}x_0=0,\ \ z_0=0\\ s_i=sg(x-x_i)\\ x_{i+1}=x_i+s_i*2^{-i}\\ z_{i+1}=z_i-s_i*2^{-i}*y\quad i=1,2,\cdots\cdots\end{cases}\tag{1}$$

When *n* is big enough, there are $x_{n+1}\approx x$, $z_{n+1}\approx xy$.

5° and 6° of Subprogram1 UV(u,v,ε) in the description of algorithm is based on iterative process (1).

Theorem1: Let $\{\delta_i\}_0^{+\infty}=\{2^{-i}\}_0^{+\infty}$. If $x\in[-R(\delta),R(\delta)]=[-2,2]$ then,

(a) $\{x_i\}$ defined by iterative process (1) converges to *x* and there is $|x-x_N|\le\delta_{N-1}$.

(b) $\{z_i\}$ defined by iterative process (1) converges to xy and there is an error estimation, $|z_N-xy|\le|y|(1+|x|)(1+\delta_N)\delta_N$.

(c) $\{z_i\}$ in Subprogram1 UV(u,v,ε) converges to xy. Its calculation error is not more than ε.

Proof: (a) From iterative process (1) it is easy to know that $x_N-x_{N+k}=\sum_{j=N}^{N+k-1}S_j\delta_j$.

From Lemma 1 in (Feng, 2006). there is

$$|x_N-x_{N+k}|=\sum_{j=N}^{N+k-1}\delta_j\le\sum_{j=N}^{\infty}\delta_j\le\delta_{N-1}$$

$\therefore\{x_i\}$ is a Cauchy series. From construct of algorithm there must be $x_i\to x\,(\mathrm{i}\to\infty)$.

Let $k\to\infty$ there is $|x-x_N|\le\delta_{N-1}$.

(b) Let *x*'s measurement expansion with $\{\delta_i\}_0^{\infty}$ be $x=\sum_{i=0}^{\infty}s_i\delta_i$ [4], and $\left\|(a_{ij})_{n\times n}\right\|_{\infty}=\max_{1\le i\le n}\sum_{j=1}^{n}|a_{ij}|$.

For $A=\begin{pmatrix}0 & 0\\ 1 & 0\end{pmatrix}$, $\|A\|_\infty = 1$.

$$\left\|e^{Ax}\right\|_\infty = \left\|\begin{pmatrix}1 & 0\\ x & 1\end{pmatrix}\right\|_\infty = 1+|x|.$$

Because N>0, from Lemma 1 in [4] there is

$$|z_N - xy| \le \left\| e^{Ax}\begin{pmatrix}y\\0\end{pmatrix} - e^{A\sum_{i=0}^{N} s_i\delta_i}\begin{pmatrix}y\\0\end{pmatrix}\right\|_\infty$$

$$\le \left\| e^{Ax} - e^{A\sum_{i=0}^{+N} s_i\delta_i}\right\|_\infty \left\|\begin{pmatrix}y\\0\end{pmatrix}\right\|_\infty = \left\|\begin{pmatrix}y\\0\end{pmatrix}\right\|_\infty \left\|e^{Ax}\right\|_\infty \left\| I - e^{-A\sum_{i=N+1}^{+\infty} s_i\delta_i}\right\|_\infty$$

$$= \left\|\begin{pmatrix}y\\0\end{pmatrix}\right\|_\infty \left\|e^{Ax}\right\|_\infty \left\| I - \left(I + \sum_{k=1}^{+\infty} \frac{\left(-A\sum_{i=N+1}^{+\infty} s_i\delta_i\right)^k}{k!}\right)\right\|_\infty$$

$$= \left\|\begin{pmatrix}y\\0\end{pmatrix}\right\|_\infty \left\|e^{Ax}\right\|_\infty \left\| A\cdot \sum_{i=N+1}^{+\infty} s_i\delta_i \cdot \sum_{k=0}^{+\infty} \frac{1}{k+1}\frac{\left(-A\sum_{i=N+1}^{+\infty} s_i\delta_i\right)^k}{k!}\right\|_\infty$$

$$\le \left\|\begin{pmatrix}y\\0\end{pmatrix}\right\|_\infty \left\|e^{Ax}\right\|_\infty \|A\|_\infty \left|\sum_{i=N+1}^{+\infty} s_i\delta_i\right| \left\| e^{\left|A\sum_{i=N+1}^{+\infty} s_i\delta_i\right|}\right\|_\infty$$

$$= \left\|\begin{pmatrix}y\\0\end{pmatrix}\right\|_\infty \left\|e^{Ax}\right\|_\infty \|A\|_\infty \sum_{i=N+1}^{+\infty}\delta_i \cdot \left\| e^{\left|\sum_{i=N+1}^{+\infty} s_i\delta_i\right|A}\right\|_\infty$$

$$= |y|(1+|x|)\cdot \sum_{i=N+1}^{+\infty}\delta_i \cdot \left(1+\left|\sum_{i=N+1}^{+\infty} s_i\delta_i\right|\right)$$

$$\le |y|(1+|x|)\cdot(1+\delta_N)\cdot\delta_N$$

$$< |y|(1+|x|)\cdot\delta_{N-1}.$$

(c) If *u*=0 or *v*=0 procedure just simply output result *uv*=0.

When $uv \neq 0$, *u* and *v* were pre-processed as $uv = s\times 2^m \times u_1 v_1$ in 2° and 3° of Subprogram1 UV (u, v, ε). Here is 1 or -1. u_1 and v_1 were limited in (0,1]. From (b), for result z'_N of 6° there is

$$|z'_N - u_1 v_1| < |u_1|(1+|v_1|)\cdot\delta_{N-1} \le 2\delta_{N-1} < \delta_{N-2}$$

In 4° of the procedure, iterative number N was set to ensure $\delta_{N-2} \le 2^{-m}\times\varepsilon$. So for last result $z_N = s\times 2^m \times z'_N$ there is

$$|z_N - uv| < 2^m \times \delta_{N-2} \le \varepsilon.$$

Proven.

Theorem 2: $B_j^n(t)$ $(j=0,...,n)$ in Subprogram2 Bernstein (n, t, ε) are calculated values of Bernstein polynomials. The upper bound of calculation error is **ε**.

Proof: We use inductive reasoning to prove result. For *n*=1,

$$B_{-1}^0(t)=0, B_0^0(t)=1, B_1^0(t)=0.$$

$$B_{-1}^1(t)=0,$$

$$B_0^1(t) = UV(1-t, B_0^0(t), \varepsilon_1) + UV(t, B_{-1}^0(t), \varepsilon_1),$$

$$B_1^1(t) = UV(1-t, B_1^0(t), \varepsilon_1) + UV(t, B_0^0(t), \varepsilon_1),$$

$$B_2^1(t)=0.$$

From procedure we know $\varepsilon_1 \le \frac{\varepsilon}{2n}$. So calculation errors of $B_0^1(t)$ and $B_1^1(t)$ are not more than $2\times\varepsilon_1 \le \frac{\varepsilon}{n}$. Inductive assuming calculation errors of $B_j^{i-1}(t)$ (*j*=0,…,*i*-1) are not more than $\frac{(i-1)\varepsilon}{n}$. Let real values of Bernstein polynomials be $b_j^i(t)$, real calculation error of $UV(1-t, B_j^{i-1}(t), \varepsilon_1)$ be ε_j. There is,

$$\left|B_j^i(t) - b_j^i(t)\right| =$$

$$\left|UV\left(1-t, B_j^{i-1}(t), \varepsilon_1\right) + UV\left(t, B_{j-1}^{i-1}(t), \varepsilon_1\right) - (1-t)b_j^{i-1}(t) - tb_{j-1}^{i-1}(t)\right|$$

$$= \left|(1-t)B_j^{i-1}(t) + \varepsilon_j + tB_{j-1}^{i-1}(t) + \varepsilon_{j-1} - (1-t)b_j^{i-1}(t) - tb_{j-1}^{i-1}\right|$$

$$\le (1-t)\left|B_j^{i-1}(t) - b_j^{i-1}(t)\right| + t\left|B_{j-1}^{i-1}(t) - b_{j-1}^{i-1}\right| + 2\times\frac{\varepsilon}{2n}$$

$$\le (1-t)\frac{(i-1)\varepsilon}{n} + t\frac{(i-1)\varepsilon}{n} + \frac{\varepsilon}{n}$$

$$= \frac{i\varepsilon}{n} \qquad \text{j=0,1,…,i.}$$

That means the upper bound of calculation errors of $B_j^n(t)$ (j=0,1, … ,n) is ε. Result proved.

Theorem 3: The upper bound of calculation error of

Table 1. Calculated values of the cubic Bezier curve.

t	Bernstein polynomials of degree 3				The cubic Bezier curve	
	$B_0^3(t)$	$B_1^3(t)$	$B_2^3(t)$	$B_3^3(t)$	*X(t)*	*Y(t)*
0.0	1.0000000075	0	0	0	0.300000003	0.300000005
0.1	0.7290000054	0.2429999958	0.0269999989	0.0009999999	0.332900002	0.375600005
0.2	0.5120000014	0.3839999993	0.0959999994	0.0079999999	0.371200002	0.424800001
0.3	0.3429999989	0.4410000002	0.1890000007	0.0270000002	0.414300002	0.451200003
0.4	0.2159999976	0.4319999992	0.2880000021	0.0640000011	0.461600002	0.458400001
0.5	0.1250000279	0.3750000084	0.3750000084	0.1250000028	0.512500013	0.450000021
0.6	0.0640000011	0.2880000021	0.4319999992	0.2159999976	0.566400000	0.429600001
0.7	0.0270000002	0.1890000007	0.4410000002	0.3429999989	0.622700002	0.400800003
0.8	0.0079999999	0.0959999994	0.3839999993	0.5120000014	0.680800003	0.367200000
0.9	0.0009999999	0.0269999989	0.2429999958	0.7290000054	0.740100005	0.332400003
1.0	0	0	0	1.0000000075	0.800000007	0.300000005

bst in Subprogram3 BS ($t, P_{i=0\ldots n}^u, \varepsilon$) is ε.

Proof: Let real values of Bernstein polynomials be $b_j^i(t)$, real value of Bezier spline be $bs(t)=\sum_{i=0}^{n} P_i^u b_i^n(t)$, real calculation error of $UV(P_i^u, B_i^n(t), \varepsilon_2)$ be ε_i, $0<\varepsilon_i\le\varepsilon_2$. Because $\varepsilon_1 \le \dfrac{\varepsilon}{\sum_{i=0}^{n}\left|P_i^u\right|+1}$ and $\varepsilon_2 \le \dfrac{\varepsilon}{(n+1)\left(\sum_{i=0}^{n}\left|P_i^u\right|+1\right)}$, we have,

$$\left|bst-bs(t)\right| = \left|\sum_{i=0}^{n} UV(P_i^u, B_i^n(t), \varepsilon_2) - \sum_{i=0}^{n} P_i^u b_i^n(t)\right|$$

$$= \left|\sum_{i=0}^{n}\left(P_i^u B_i^n(t)+\varepsilon_i\right) - \sum_{i=0}^{n} P_i^u b_i^n(t)\right|$$

$$= \left|\sum_{i=0}^{n} P_i^u B_i^n(t) + \sum_{i=0}^{n}\varepsilon_i - \sum_{i=0}^{n} P_i^u b_i^n(t)\right|$$

$$\le \left|\sum_{i=0}^{n} P_i^u B_i^n(t) - \sum_{i=0}^{n} P_i^u b_i^n(t)\right| + (n+1)\varepsilon_2$$

$$\le \sum_{i=0}^{n}\left|P_i^u\right|\left|B_i^n(t)-b_i^n(t)\right| + (n+1)\frac{\varepsilon}{(n+1)\left(\sum_{i=0}^{n}\left|P_i^u\right|+1\right)}$$

$$\le \left(\sum_{i=0}^{n}\left|P_i^u\right|\right)\varepsilon_1 + \frac{\varepsilon}{\sum_{i=0}^{n}\left|P_i^u\right|+1}$$

$$\le \left(\sum_{i=0}^{n}\left|P_i^u\right|\right)\frac{\varepsilon}{\sum_{i=0}^{n}\left|P_i^u\right|+1} + \frac{\varepsilon}{\sum_{i=0}^{n}\left|P_i^u\right|+1} = \varepsilon$$

Result proved.

Theorem 4: The upper bound of calculation error of Bezier curves with this algorithm is $\sqrt{2}\varepsilon$.

Proof: Let $BSx(t)=\sum_{i=0}^{n} P_i^x b_i^n(t)$ and $BSy(t)=\sum_{i=0}^{n} P_i^y b_i^n(t)$ denote *x* value and *y* value of Bezier curve. From Theorem 3, both values' calculation error are not more than ε, then the overall error is not more than of $\sqrt{\varepsilon^2+\varepsilon^2}=\sqrt{2}\varepsilon$. Proven.

NUMERICAL EXPERIMENT

Experiment 1: Given 4 control points (0.3, 0.3), (0.4, 0.6), (0.6, 0.4), (0.8, 0.3) and the upper bound of error $\varepsilon = 5\times10^{-7}$. Using algorithm presented in this paper to compute a cubic Bezier curve. Results are shown below. Iteration of Subprogram1 UV(u, v, ε) is not more than 28 steps.

Real velues of Bernstein polynomials and the cubic Bezier curve are shown in . Tables 1, 2 and Figure 1,

Experiment 2: For another set of control points, (0, 0.8), (0.3, 0.4),(0.6, 0.2), (0.9, 0.6), and the upper bound of

Table 2. Real values of the cubic Bezier curve.

t	Bernstein polynomials of degree 3				The cubic Bezier curve	
	$B_0^3(t)$	$B_1^3(t)$	$B_2^3(t)$	$B_3^3(t)$	*X(t)*	*Y(t)*
0.0	1	0	0	0	0.3	0.3
0.1	0.729	0.243	0.027	0.001	0.3329	0.3756
0.2	0.512	0.384	0.096	0.008	0.3712	0.4248
0.3	0.343	0.441	0.189	0.027	0.4143	0.4512
0.4	0.216	0.432	0.288	0.064	0.4616	0.4584
0.5	0.125	0.375	0.375	0.125	0.5125	0.45
0.6	0.064	0.288	0.432	0.216	0.5664	0.4296
0.7	0.027	0.189	0.441	0.343	0.6227	0.4008
0.8	0.008	0.096	0.384	0.512	0.6808	0.3672
0.9	0.001	0.027	0.243	0.729	0.7401	0.3324
1.0	0	0	0	1	0.8	0.3

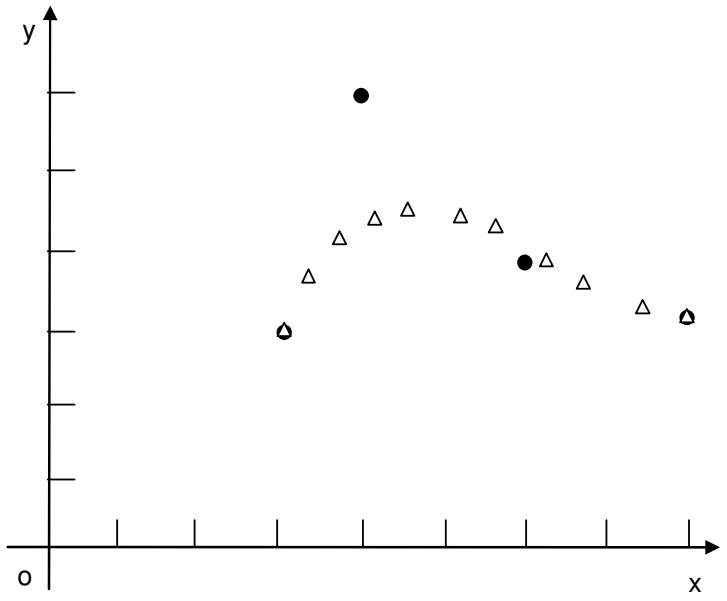

Figure 1. Data points of the cubic Bezier curve.

error $\varepsilon = 5\times10^{-7}$, computing results of a cubic Bezier curve are shown in Table 3 and Figure 2.

Results for Bernstein polynomials of degree 3 are the same as those in experiment 1. Iteration of Subprogram1 UV(u, v, ε) is not more than 27 steps.

Experiment 3: For control points (0, 0.8), (0.5, 0.3), (0.9, 0.6) and the upper bound of error $\varepsilon = 5\times10^{-7}$, generate a quadratic Bezier curve. Calculated values and real values of both Bernstein polynomials and the Bezier curve are shown in Table 4 and Figure 3. Iteration of Subprogram1 UV (u, v, ε) is not more than 25 steps. Real velues of Bernstein polynomials of degree 2 and the quadratic Bezier curve are shown in Table 5 and Figure 3, all calculation errors are under control.

Conclusion

Above are some discussions for a shift-add algorithm based on CORDIC algorithm for computing Bezier curves. This algorithm can be implemented in basic computing system which exists in many areas.

Theoretical analysis showed the convergence of the algorithm. Error estimations were provided. A numerical experiment was carried out to validate the algorithm's effectiveness and efficiency. We can say this is a fast,

Table 3. Calculated values and real values of the cubic Bezier curve.

t	Calculted values		Real values	
	X(t)	Y(t)	X(t)	Y(t)
0.0	0	0.800000015	0	0.8
0.1	0.090000034	0.686400012	0.09	0.6864
0.2	0.179999988	0.587799971	0.18	0.5872
0.3	0.269999984	0.504799997	0.27	0.5048
0.4	0.359999997	0.441600026	0.36	0.4416
0.5	0.450000046	0.400000053	0.45	0.4
0.6	0.540000019	0.382400008	0.54	0.3824
0.7	0.630000006	0.391199973	0.63	0.3912
0.8	0.719999982	0.428799986	0.72	0.4288
0.9	0.810000010	0.497600000	0.81	0.4976
1.0	0.900000011	0.600000007	0.9	0.6

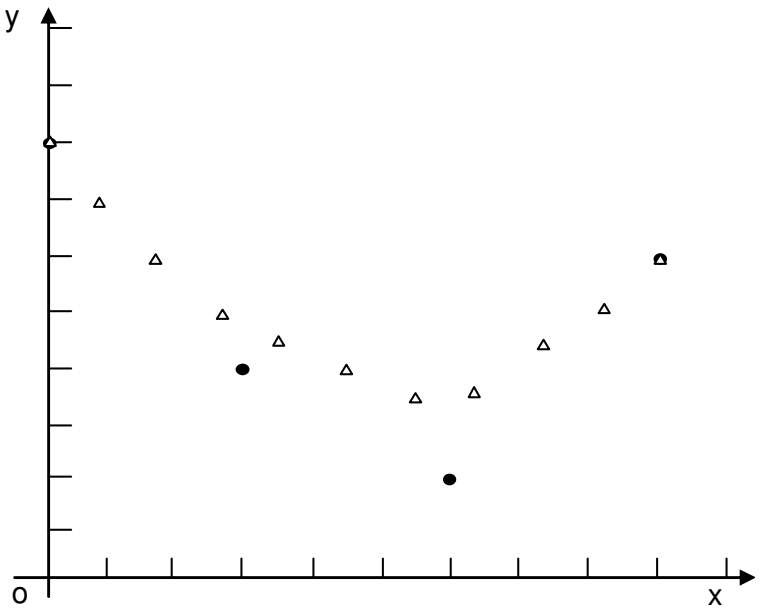

Figure 2. Data points of the cubic Bezier curve.

Table 4. Calculated values of the quadratic Bezier curve.

t	Bernstein Polynomials of degree 2			The quadratic Bezier curve	
	$B_0^2(t)$	$B_1^2(t)$	$B_2^2(t)$	*X(t)*	*Y(t)*
0.0	1.000000030	0	0	0	0.8000000060
0.1	0.809999984	0.180000040	0.010000005	0.0990001645	0.7080000041
0.2	0.639999995	0.319999983	0.039999995	0.1959999453	0.6319999767
0.3	0.490000025	0.420000018	0.090000007	0.2910000278	0.5720000536
0.4	0.369999993	0.480000002	0.160000005	0.3840000637	0.5280000117
0.5	0.250000030	0.500000060	0.250000030	0.4750000805	0.5000000894
0.6	0.160000005	0.480000002	0.359999993	0.5640000482	0.4880000045
0.7	0.090000007	0.420000018	0.490000025	0.6510000743	0.4920000095
0.8	0.039999995	0.319999983	0.639999995	0.7359999453	0.5119999767
0.9	0.010000005	0.180000040	0.809999984	0.8190001329	0.5480000255
1.0	0	0	1.000000030	0.9000000328	0.6000000119

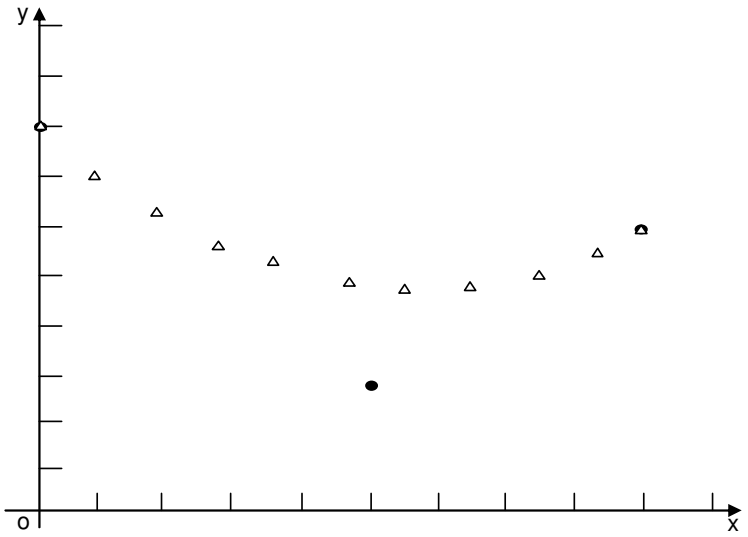

Figure 3. Data points of the quadratic Bezier curve.

Table 4. Calculated values of the quadratic Bezier curve.

t	Bernstein Polynomials of degree 2			The quadratic Bezier curve	
	$B_0^2(t)$	$B_1^2(t)$	$B_2^2(t)$	***X(t)***	***Y(t)***
0.0	1.000000030	0	0	0	0.8000000060
0.1	0.809999984	0.180000040	0.010000005	0.0990001645	0.7080000041
0.2	0.639999995	0.319999983	0.039999995	0.1959999453	0.6319999767
0.3	0.490000025	0.420000018	0.090000007	0.2910000278	0.5720000536
0.4	0.369999993	0.480000002	0.160000005	0.3840000637	0.5280000117
0.5	0.250000030	0.500000060	0.250000030	0.4750000805	0.5000000894
0.6	0.160000005	0.480000002	0.359999993	0.5640000482	0.4880000045
0.7	0.090000007	0.420000018	0.490000025	0.6510000743	0.4920000095
0.8	0.039999995	0.319999983	0.639999995	0.7359999453	0.5119999767
0.9	0.010000005	0.180000040	0.809999984	0.8190001329	0.5480000255
1.0	0	0	1.000000030	0.9000000328	0.6000000119

Table 5. Real values of the quadratic Bezier curve.

t	Bernstein Polynomials of degree 2			The quadratic Bezier curve	
	$B_0^2(t)$	$B_1^2(t)$	$B_2^2(t)$	***X(t)***	***Y(t)***
0.0	1	0	0	0	0.8
0.1	0.81	0.18	0.01	0.099	0.708
0.2	0.64	0.32	0.04	0.196	0.632
0.3	0.49	0.42	0.09	0.291	0.572
0.4	0.36	0.48	0.16	0.384	0.528
0.5	0.25	0.5	0.25	0.475	0.5
0.6	0.16	0.48	0.36	0.564	0.488
0.7	0.09	0.42	0.49	0.651	0.492
0.8	0.04	0.32	0.64	0.736	0.512
0.9	0.01	0.18	0.81	0.819	0.548
1.0	0	0	1	0.9	0.6

effective and efficient algorithm and could be of many applications.

REFERENCES

Aumann G (1997). Corner cutting curves and a new characterization of Bezier and B-spline curves. Computer Aided Geometric Design, 14(5):449-474.

Catherine C, Sonya S (2003). Circular Bernstein--Bézier spline approximation with knot removal. J. Comput. Appl. Math., 155(1):177-185.

Farin G (1997). Curves and Surfaces for Computer-aided Geometric Design: A Practical Guide, 4th Ed. Academic Press, San Diego.

Feng G (2006). Convergence and Error Estimation of Coordinate Rotating Algorithm and Its Expansion. Chn. J. Num. Math. Applications. 28(2):1-9.

Han X, Ma Y, Huang X (2008). A novel generalization of Bezier curve and surface. J. Comput. Appl. Math., 217(1):180-193.

Muller JM (2006). Elementary Functions, Algorithms and Implementation. Birkhauser Boston. 1st edition, 1997. 2nd edition, pp.133-156.

Neil Eklund (1998). CORDIC: Elementary Function Computation Using Recursive Sequences. International Conference on Technology in Collegiate Mathematics (ICTCM). p. 11.

Popiel T, Noakes L (2006). C-2 spherical Bezier splines. Computer Aided Geometric Design., 23(3):261-275.

Ray Andraka (1998). A Survey of CORDIC Algorithms for FPGA Based Computers. In Proceedings of the 1998 ACM/SIGDA Sixth International Symposium on Field Programmable Gate Arrays(FPGA) pp.191-200

Volder JE (1959). The CORDIC Computing Technique. IRE Transactions on Electronic Comput., 8(9):330-334.

Xiaobo H, Ronald H, Steven B (1991). Expanding the Range of Convergence of the CORDIC Algorithm. IEEE Transactions on Comput., 40(1):13-21.