International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:2, No:1, 2008

Optimal Straight Line Trajectory Generation in 3D Space using Deviation Algorithm

T.C.Manjunath *Ph.D.* (*IIT Bombay*) & *Fellow IETE* , C. Ardil

***Abstract*—**This paper presents an efficient method of obtaining a straight-line motion in the tool configuration space using an articulated robot between two specified points. The simulation results & the implementation results show the effectiveness of the method.

***Keywords*—**Bounded deviation algorithm, Straight line motion, Tool configuration space, Joint space, TCV.

I. Introduction

STRAIGHT line motion is defined as the motion along a straight line or movement of a rigid body along a straight line and represents the shortest distance between the two points in the 3D workspace of any robot. The straight line motion from the source (pick) to the goal (place) covered in a specific amount of time is known as the straight line trajectory, i.e., if temporal information is added to the straight line path by specifying the times at where the gripper or tool-tip is along the straight line path, then the straight line path gets converted into a straight line trajectory [1].

Straight-line motion is always required in TCS R^6. By controlling all the joints in a coordinated manner, the tool-tip can be made to move along a straight-line path. If the distance between the adjacent points in the joint space R^n is approximately small, then a straight-line path or trajectory in the TCS R^6 can be designed. How we get straight-line motion is to use the IK equations. In these paragraphs, we give info about the trajectory in joint space, which generates a straight-line trajectory in TCS by using the IK equations [3].

The applications of straight line motions are listed as follows.

(a) Conveyor belt operations.
(b) Straight line seam arc welding.
(c) Inserting peg into a hole.
(d) Threading a nut onto a bolt.
(e) Performing screw transformations.
(f) For inserting electronic components onto PCB.
(g) Doing robotic manipulation from above the object (used where exact perpendicularity is required).
(h) Inspection of manufactured components which are coming on a conveyor belt (using computer / robot vision).

T.C. Manjunath, a Ph.D. from IIT Bombay is currently, Professor & Head in Electronics and Communications Engg. Dept. of New Horizon College of Engg., Bangalore-87, Karnataka, India.
E-mail: tcmanjunath@rediffmail.com ; tcmanjunath@gmail.com.

C. Ardil is with the National Academy of Aviation, AZ 1056 Baku, Azerbaijan.

The paper is organized in the following sequence. A brief introduction about the straight line motion was presented in the previous paragraphs along with the applications. A review of the straight line motion theory is presented in the section 2. The bounded deviation algorithm used in the paper is discussed in section 3. A mathematical formulation of the simulation study is depicted in section 4 followed by the simulation results in section 5. Section 6 gives the .C code used to develop the straight line path. This section is followed by the conclusions in section 7 and then the references.

II. Review of Straight Line Motion Theory

Consider the Fig. 1. Let w^0 and w^1 be the two points in the space between which the robot has to draw a straight line. Here, we use the following parameters as [2];

w^0 : is the source point (initial point) ; i.e., the TCV at the point 0 ;
w^1 : is the goal point (final point) ; i.e., the TCV at the point 1 ;
w^0 and w^1 : both are (6×1) vectors in TCS, R^6.
T : is the total time taken to move from w^0 to w^1, i.e., the total time taken to traverse the path, obviously $T > 0$.
Γ = $\{ w^0, w^1 \}$ = path taken by tool.

The equation for SL path / straight line trajectory w(t) of the tool as shown in the Fig. 1 is represented by an equation of 1st degree or of 1st order, i.e., no squared terms in the expression, i.e., we are writing an expression for the straight line path or trajectory in terms of the SDF and the TCV [2].

$$w(t) = [\, 1 - S(t) \,]\, w^0 + s(t)\, w^1 \quad ; 0 \le t \le T \qquad (1)$$

where s(t) is a differentiable Speed Distribution mapping Function [SDF] which maps (0 , T) into (0 , 1) & is given by [2]

$$s(t) = \frac{t}{T} \qquad (2)$$

At the start of the trajectory, t = 0 ;
i.e., s (0) = 0 (start ; initial ; pick ; source point)

$$\therefore, \quad w(t) = [\, 1 - 0 \,]\, w^0 + 0\, w^1 = w^0 \qquad (3)$$

..... corresponds to start of the path [2].

At the end of the trajectory, t = T,
i.e., s (T) = 1

i.e., the end ; goal ; destination ; place ; final point.

$\therefore$, $w(t) = [\ 1 - 1\ ]\ w^0 + 1\ w^1 = w^1$ (4)

..... corresponds to end of the path [2].

Hence, it is verified that the Eq^n (1) is of the first order or first degree, i.e., a straight line equation of the form y = mx + c, where c is the intercept, m is the slope.

III. Bounded Deviation Algorithm [BDA] and its Basic Working Principle

BDA is an algorithm, which is used to obtain an approximated straight-line motion in TCS R^6 by using an articulated robot by selecting the number of knot points properly [4], minimizing them and distributing them along the trajectory in an optimal manner [2].

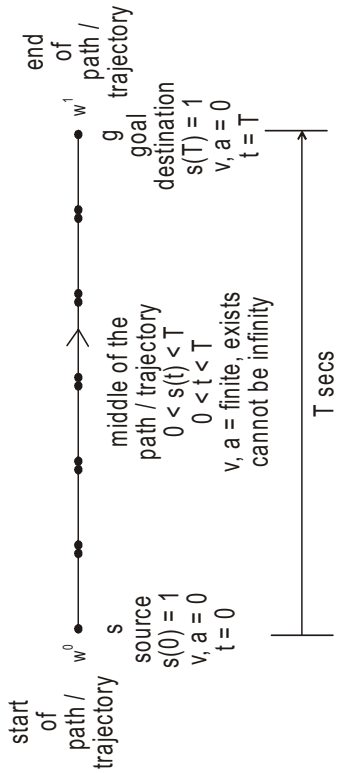

Fig. 1 Straight line motion, a graphical representation

A. Principle of BDA

It is easier to produce a straight line motion (SL path or SL trajectory) in case of xyz, PTP, cylindrical, polar or spherical or SCARA or Stanford robots. But, in case of articulated robots, it is very difficult to obtain a straight line motion in the TCS. All the joints has to be activated simultaneously in a coordinated manner in order to make the tool-tip to move in a straight line. For achieving a straight line motion in the TCS, the following procedure is used [2].

A straight line path in joint space R^n will not produce a straight line path in tool configuration space R^6. Hence, the straight line path in TCS is obtained by using approximation techniques using an algorithm called as the Bounded Deviation Algorithm [BDA].

BDA was proposed by Taylor and it gives the deviation or the error between the actual straight line trajectory and the trajectory generated by straight line motion in joint space. Deviation is likely to be maximum somewhere near the mid-point of the joint space trajectory [5]. Check the error or deviation at this mid-point of the joint space trajectory [2].

If it exceeds a prescribed tolerance limit, then the exact mid-point is added as knot point. Repeat the test recursively on the newly generated segments until all the knot points and the mid point deviations are within tolerance limit. If the error or deviation is minimum and bounded (within the tolerance limit of ε) ; then, we get a approximated straight line path between w^0 and w^1 [6].

If distance between adjacent points in JS (joint space) is small, a straight line path segment in JS can be approximated to a straight line path in TCS. Therefore, we can approximate a straight line path by visiting a number of closely spaced knot points in proper sequence in joint space as shown in Fig. . The result is the straight line trajectory in TCS. Since there is no direct control over the tool-tip and the only thing that we can control is the joint (since motor / piston is connected to it), an approximated joint space trajectory can be used to obtain a straight line trajectory in the TCS [7].

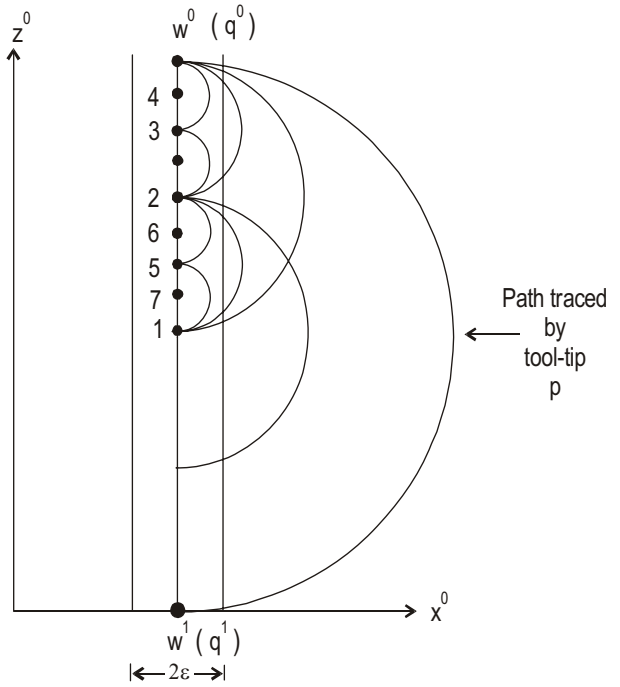

Fig. 2 Interpolation of joint space approximation to the straight line motion

Inverse kinematics equations [2] have to be solved at each point after minimizing the number of knot points on the trajectory and distributing them along the trajectory in an optimal manner. The flow-chart of the BDA algorithm used is shown in the Fig. 3.

B. Bounded Deviation Algorithm for Obtaining Straight Line Motion [2]

1. Select a tolerance limit (called as threshold value) for straight line motion as $\varepsilon > 0$ [8].
2. Given, the start point and end point of trajectory as w^0 and w^1 [TCV's at the starting and ending points], use the inverse kinematics equations to compute q^0, q^1 ; i.e., the joint vectors associated with $\{w^0, w^1\}$.
   $w^0 \rightarrow \text{IKP} \rightarrow q^0$;
   $w^1 \rightarrow \text{IKP} \rightarrow q^1$.
3. Compute the joint space mid-point as
   $$q_m = \frac{q_0 + q_1}{2}$$
4. Use the information from q^m and tool configuration vector w to find the equivalent TC space mid-point as
   $w^m = w(q^m)$
   i.e., using q^m, find the TCV, $w(q^m)$; substitute q^m in the TCV of that particular robot which is used to obtain the straight line motion and obtain w^m.
5. Find the exact TCS mid-point as
   $$w^M = \frac{w^0 + w^1}{2}$$
6. If the error or deviation $\| w^m - w^M \| \leq \varepsilon$; then, stop.
7. Else, insert w^M as a exact knot point between w^0 and w^1. Now, the trajectory is broken up into two parts, viz., $\{w^0, w^M\}$ and $\{w^M, w^1\}$.
8. Repeat the steps (1 to 6) recursively to the newly generated trajectory segments $\{w^0, w^M\}$ and $\{w^M, w^1\}$ till all the newly generated trajectory segments are within limit of ε.

Bounded deviation algorithm does not distribute knot points uniformly over the interval $\{w^0, w^1\}$. Instead, it places where they are most needed to reduce the deviation between the exact trajectory and the joint space trajectory [9]. Distribution of knot points is not uniform and depends on ε, geometry of robot, the limitations of the joints and its constraints, straight line path and location, w^0 and w^1 [2].

Referring to the Fig. 2, we get the joint space interpolated motion between $\{w^0, w^1\}$ deviates outside the cylinder of radius ε [10]. Therefore, insert knot point 1 as mid-point. Again, joint space interpolated motion between $\{w^0, 1\}$ is outside the cylinder of radius ε. Therefore, insert knot point 2 as mid-point [2].

Again, joint space interpolated motion between $\{w^0, 2\}$ is outside the cylinder of radius ε. Therefore, insert knot point 3 as the mid point. Now, joint space interpolated motion between $\{w^0, 3\}$ is within the tolerance limit of ε [2].

Therefore, the path from w^0 to 3 is a approximated straight line path. Like this, continue till all the newly generated paths are within the cylinder of radius ε. If ε is very very small ($\cong 0$, i.e., 0.000001, say), then the number of knot points will be very very close ; then, the path traced by the tool-tip from w^0 to w^1 will be an exact straight line as the number of iterations will be more [2].

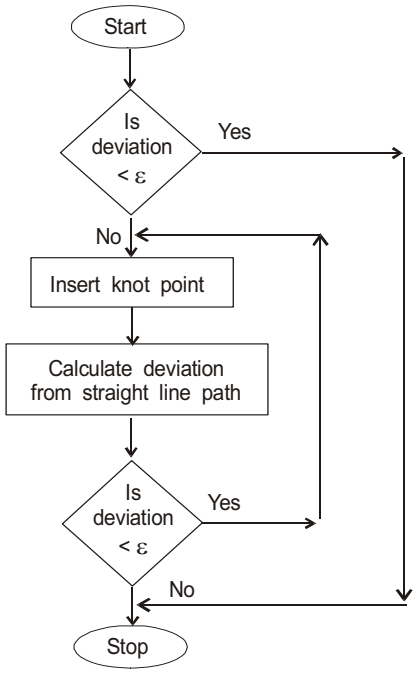

Fig. 3 BDA flow chart

IV. A Mathematical Formulation Of The Simulation Study

A simulation is performed on a five axis articulated robot which was designed and fabricated in the college laboratory as shown in Fig. 4 [11].

Fig. 4 Photographic view of the designed robot

One pass of BDA is shown analytically here to find a joint space knot point for approximating the following straight-line trajectory [13]. We consider the tool configuration vectors at the starting point and the ending point to be specified by the user as [12]

$$w^0 = [\,600\,,\ 0\,,\ 250\,,\ 0\,,\ 0\,,\ -2\,]^T$$
$$w^1 = [\,600\,,\ 0\,,\ 50\,,\ 0\,,\ 0\,,\ -2\,]^T$$

The physical dimensions of the designed system are as follows.

$d = [495.2 \quad 0 \quad 0 \quad 0 \quad 368.2\ ]^T$ and

$a = [\ 0 \quad 457.2 \quad 457.2 \quad 19 \quad 0\ ]^T$?

The tool configuration deviation at the joint-space midpoint is obtained as follows [14]. The general inverse kinematic equations [2] for any five axis articulated robot is [2]

Base angle θ_1 (rotary) :

$$q_1 = \tan^{-1}\left(\frac{w_2}{w_1}\right)$$

GTP angle θ_{234} :

$$q_{234} = \tan^{-1}\left(\frac{-(C_1 w_4 + S_1 w_5)}{-w_6}\right)$$

$$= -\arctan 2\,(-b_0, -w_6)$$

Intermediate variables are :

$$b_1 = C_1 w_1 + S_1 w_2 - a_4 C_{234} + d_5 S_{234}$$

$$b_2 = d_1 - a_4 S_{234} - d_5 C_{234} - w_3$$

Shoulder angle θ_2 (rotary) :

$$q_2 = \tan^{-1}\left(\frac{(a_2 + a_3 C_3) b_2 - (a_3 S_3) b_1}{(a_2 + a_3 C_3) b_1 + (a_3 S_3) b_2}\right)$$

Elbow angle θ_3 (rotary) :

$$q_3 = \cos^{-1}\left(\frac{b_1^2 + b_2^2 - a_2^2 - a_3^2}{2 a_2 a_3}\right)$$

Tool pitch angle θ_4 (rotary) :

$$q_4 = q_{234} - q_2 - q_3$$

Tool roll angle θ_5 (rotary) :

$$q_5 = \pi\, ln\sqrt{w_4^2 + w_5^2 + w_6^2}$$

Find the joint vectors q^0 and q^1 associated with w^0 and w^1 by using the above inverse kinematic equations.

Give w^0 as the input to IK algorithm of a five axis articulated robot and calculate q^0 [15], [22]. We get the parameters base angle, global tool pitch angle, intermediate variables, elbow angle, shoulder angle, tool pitch angle & the tool roll angle w.r.t. the TCV w^0 [2], [21].

Then, give w^1 as the input to IK algorithm of a five axis articulated robot and calculate q^1 [17], [20] We get the parameters base angle, global tool pitch angle, intermediate variables, elbow angle, shoulder angle, tool pitch angle & the tool roll angle w.r.t. the TCV w^1 [2].

Then, compute the joint space midpoint of the trajectory, i.e., q^m as the average of q^0 and q^1. The TCV $w(q)$ of a five axis articulated robot is then used to compute the tool configuration vector at the actual mid point as w^m [2].

$$w(q) = \begin{bmatrix} C_1\{(a_2C_2 + a_3C_{23}) + a_4C_{234} - d_5S_{234}\} \\ S_1\{(a_2C_2 + a_3C_{23}) + a_4C_{234} - d_5S_{234}\} \\ d_1 - (a_2S_2 + a_3S_{23}) - a_4S_{234} - d_5C_{234} \\ \cdots\cdots\cdots\cdots \\ -\left\{\exp\left(\frac{q_5}{\pi}\right)\right\} C_1S_{234} \\ -\left\{\exp\left(\frac{q_5}{\pi}\right)\right\} S_1S_{234} \\ -\left\{\exp\left(\frac{q_5}{\pi}\right)\right\} C_{234} \end{bmatrix}$$

$q_1 = 0°$

$q_{234} = 0°$

$b_1 = 581$

$b_2 = 58.4$

$q_3 = \pm 201.24°$

$q_2 = -89.14$ (assume $q_3 = +\ 201.24°$)

$q_4 = -118.08°$

$q_5 = 0°$

$w^0 \rightarrow IKP \rightarrow q^0 = [\ 0 \quad -89.14 \quad 201.24 \quad -118.08 \quad 0\ ]^T$ degs

Give w^1 as the input to IK algorithm of a five axis articulated robot and calculate q^1 [2].

$q_1 = 0°$

$q_{234} = 0°$

$b_1 = 581$

$b_2 = -141.6$

$q_3 = \pm 196.6°$

$q_2 = -127.7$ (assume $q_3 = +\ 196.6°$)

$q_4 = -71°$,

$q_5 = 0°$

$w^1 \rightarrow IKP \rightarrow q^1 = [\ 0 \quad -125.6 \quad 196.6 \quad -71 \quad 0\ ]^T$ degs

The joint space midpoint of trajectory is [16]

$$q_m = \frac{q^0 + q^1}{2} = [\ 0 \quad -107.2 \quad 198.8 \quad -91.6 \quad 0\ ]^T \text{ degs}$$

The TCV of a five axis articulated robot is

Putting the value of q^m, d_1, d_5, a_2, a_3, a_4 in $w(q)$, we get

$w^m = [\ 609 \quad 0 \quad 148.8 \quad 0 \quad 0 \quad -2\ ]^T = w(q^m)$

The exact TCS midpoint is

$$w^M = \frac{w^0 + w^1}{2} = [\ 600 \quad 0 \quad 150 \quad 0 \quad 0 \quad -2\ ]^T$$

$\therefore$, the midpoint deviation between w^m and w^M is

$w^m - w^M = [\ 9 \quad 0 \quad -1.2 \quad 0 \quad 0 \quad 0\ ]^T$

$||w^m - w^M|| = 9.08$

Since the deviation is very much greater than the threshold value ε, which is normally very small, insert w^M as a exact know point [18].

Now, the trajectory is divided into (w^0,w^M) and (w^M, w^1). Again, using w^M as the input to the IK algorithm (use the five IK equations), we get the joint angle vector q^M at the mid point as [19]

$q^M = [\ 0 \quad -109 \quad 201.8 \quad -92.8 \quad 0\ ]^T$ degs

V. Simulation Results

A graphical user interface program in C / C++ is developed and the simulation results are shown in the Figs. 5 to 8 respectively.

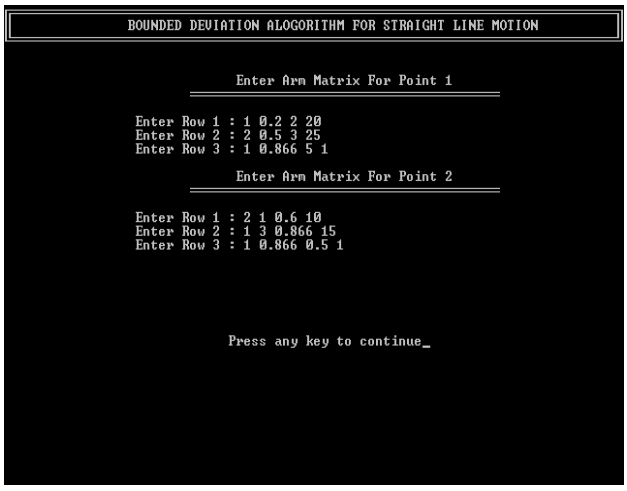

Fig. 5 Input vectors to the algorithm

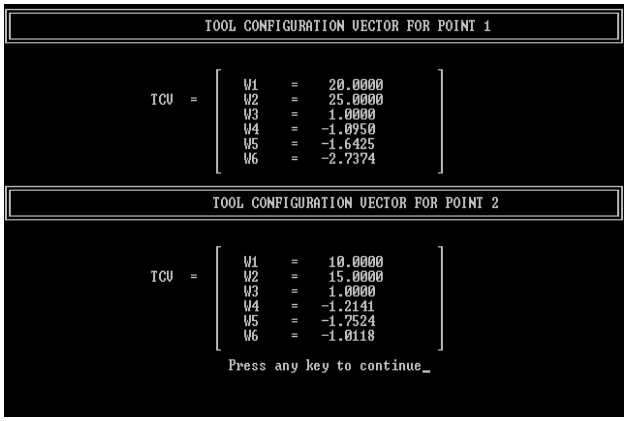

Fig. 6 Tool configuration vectors of the algo

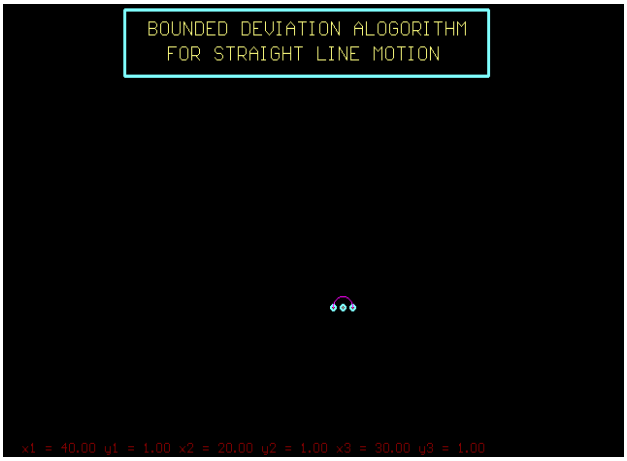

Fig. 7 One pass of the algorithm

Fig. 8 Another pass of the algorithm

VI. Generation of the .C Code

```
/*********************************************
DESCRIPTION: This program is designed to do bounded
deviation algorithm for 5 axis articulated robot. It calculates
the TCV. It also shows the final arm matrix & the shortest
path between two points in joint space.
*********************************************

#include <iostream.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <graphvar.h>
#include<graphics.h>

const int DOF = 5;
const double PI = 4 * atan(1.0);
const char wait[] = "Press any key to continue";
const int PAUSE = 1;//indicates function should pause for
input
```

```
class Hctm          //homogeeous coordiate trasformatio matrix
{
  private : double mat[4][4];

  public :  Hctm();
        Hctm(double m[ ][4]);
        void initLctm(double ang[ ],double d[ ],double a[
],double alp[ ],int k);
        void initIdentity();//Initialise as identity matrix
        void printLctm(int k,int k1);
        void getvalues(double arm[][4]);
        Hctm operator * (Hctm a);
};

Hctm::Hctm()
{
 int i,j;
 for(i = 0;i < 4;i++)
     for(j = 0;j < 4;j++) mat[i][j] = 0;

}

Hctm::Hctm(double a[][4])
{
 int i,j;
 for(i = 0;i < 4;i++)
     for(j = 0;j < 4;j++)
   mat[i][j] = a[i][j];
}

void Hctm::initIdentity()
{
  for(int i = 0;i < 4;i++)
    mat[i][i] = 1;
}

void Hctm::getvalues(double arm[][4])
{
  int i,j;

  for(i = 0;i < 4;i++)
   for(j = 0;j < 4;j++)
       arm[i][j] = mat[i][j];

}

void Hctm::initLctm(double ang[],double d[],double
a[],double alp[],int l)
{//Initialises mat with Lctm T(k to k-1)
 int k = l - 1;
 double co = cos(ang[k]);
 double so = sin(ang[k]);
 double ca = cos(alp[k]);
 double sa = sin(alp[k]);

 mat[0][0] = co;
 mat[0][1] = -ca * so;
 mat[0][2] = sa * so;
 mat[0][3] = a[k] * co;
 mat[1][0] = so;
 mat[1][1] = ca * co;
 mat[1][2] = -sa * co;
 mat[1][3] = a[k] * so;
 mat[2][1] = sa;
 mat[2][2] = ca;
 mat[2][3] = d[k];
 mat[3][3] = 1;
}

void Hctm::printLctm(int k,int k1)
{
  int y = wherey(),i,j,x;

  gotoxy(10,y + 2);cout<<"T  = ";
  gotoxy(11,y + 1);cout<<k;
  gotoxy(11,y + 3);cout<<k1;
  gotoxy(15,y);cout<<'Ú';
  gotoxy(15,y + 5);cout<<'À';

  for(i = 0;i < 4;i++)
  {
   gotoxy(15,y + i + 1);
   cout<<"³   ";
   for(j = 0;j < 4;j++)
      printf("%- 10.4f   ",mat[i][j]);
   x = wherex();cout<<'³';
  }
  gotoxy(x,y);cout<<'¿';
  gotoxy(x,y + 5);cout<<'Ù';
  gotoxy(1,y + 6);
}

Hctm Hctm::operator * (Hctm a)
{
 Hctm res;
 int i,j,k;
 double sum = 0;

 for(i = 0;i < 4;i++)
    for(j = 0;j < 4;j++)
    {
   sum = 0;
   for(k = 0;k < 4;k++)
      sum += mat[i][k] * a.mat[k][j];
   res.mat[i][j] = sum;
    }

 return res;
}

double * IK(double arm[][4],double a[],double d[],double
tcv[],double ang2[]);
```

```
double * IK(double ak[],double dk[],double tcv[],double
ang2[]);
double *getTCV(double arm[][4]);
Hctm DK(double a[],double d[],double alp[],double ang[]);
void BDA(double w1[],double w2[],double ak[],double
dk[],double alpha[],double ep);
void getxy(double tcv[],int &x,int &y,int bx,int by);
void getcabinetxy(double tcv[],double &x,double &y);
double getDistance(double w1[],double w2[]);
void printLctm(double mat [][4],int k,int k1);
void printvector(double an[],int len = DOF,char *m =
"Q",char sym = 'é');
inline void pause(char *s = (char *)wait);
int gpause(const char *s,int color = YELLOW,int p =
PAUSE,int y = 465);
void line(char c = 'Í',int len = 78);
void swap(double &a,double &b);

int main(void)
{
 clrscr();
 double dk[DOF] = {26.04,0,0,0,16.83};
 double ak[DOF] = {0,22.86,22.86,.95,0};
 //double dk[DOF] = {50,0,0,0,25};
 //double ak[DOF] = {0,30,30,30,0};

 double alpha[DOF] = {-PI/2,0,0,-PI/2,0};
 double epsilon = 1; //THRESHOLD VALUE ASSUMED
 double arm[4][4];
 int i,j,k;

 cout.setf(ios::stdio);
 cout<<'É';line();cout<<'»';
 cout<<"º\t       BOUNDED DEVIATION ALOGORITHM
FOR STRAIGHT LINE MOTION";
 gotoxy(80,wherey());cout<<'º';
 cout<<'È';line();cout<<"¼\n";
 double *w1 = NULL,*w2 = NULL;
 for(k = 0;k < 2;k++)
 {
    cout<<"\n\t\t\t      Enter Arm Matrix For Point "<<(k +
1)<<"\n";
    cout<<"\t\t\t";line('Í',40);cout<<"\n\n";
    for(i = 0;i < 3;i++)
    {
      cout<<"\t\t Enter Row "<<(i + 1)<<" : ";
      for(j = 0;j < 4;j++)
        cin>>arm[i][j];
    }
    for(i = 0;i < 3;i++)
     arm[3][i] = 0;
    arm[3][3] = 1;
    if (k == 0)  w1 = getTCV(arm);
    else w2 = getTCV(arm);
 }
 pause();
 clrscr();

 for(k = 0;k < 2;k++)
 {
  cout<<'É';line();cout<<'»';
  cout<<"º\t\t\t   TOOL CONFIGURATION VECTOR FOR
POINT " <<(k + 1);
  gotoxy(80,wherey());cout<<'º';
  cout<<'È';line();cout<<"¼\n";
  if (k == 0) printvector(w1,6,"TCV",'W');
   else printvector(w2,6,"TCV",'W');
 }
 pause();
 clrscr();

 int gd = DETECT,gm,gerror;
 initgraph(&gd,&gm,"c:\\tc\\bgi\\");
 if ((gerror = graphresult() ) < 0)
 {
  printf("\n\t Error in initialising graphics : %s",
  grapherrormsg(gerror));
  return 1;
 }

 char head1[] = "BOUNDED DEVIATION
ALOGORITHM";
 char head2[] = "FOR STRAIGHT LINE MOTION";
 int c,d,d1;

 cleardevice();
 setcolor(YELLOW);setbkcolor(BLACK);
 settextstyle(SMALL_FONT,HORIZ_DIR,7);
 setlinestyle(SOLID_LINE,0,THICK_WIDTH);
 c = textwidth(head1); d = textheight(head1);
 d1 = textheight(head2);
 gprintf(150,10,head1);
 gprintf(170,15 + d,head2);
 setcolor(LIGHTCYAN);rectangle(125,5,170 + c,30 + d +
d1);

settextstyle(SMALL_FONT,HORIZ_DIR,5);setcolor(LIGHT
MAGENTA);
 getch();
 BDA(w1,w2,ak,dk,alpha,epsilon);
 gpause("Press Any Key To Continue");

 return 0;
}

double *getTCV(double arm[][4])
{
 int i;
 double *tcv = new double[6];
 double a1,t1,t2,a5;

 for(i = 0;i < 3;i++)
  tcv[i] = arm[i][3];
 a1 = atan2(tcv[1],tcv[0]);
```

```
  double s1 = sin(a1);
  double c1 = cos(a1);
  t1 = s1 * arm[0][0] - c1 * arm[1][0];
  t2 = s1 * arm[0][1] - c1 * arm[1][1];
  a5 = atan2(t1,t2);

  for(i = 0;i < 3;i++)
   tcv[i + 3] = -exp(a5/PI) * arm[i][2];

  return tcv;
}

double * IK(double arm[][4],double ak[],double dk[],double
tcv[],double ang2[])
{
  int i;
  double *ang = new double[DOF];
  double t1,t2,q234,b1,b2,modb;

  for(i = 0;i < 3;i++) tcv[i] = arm[i][3];
  ang[0] = atan2(tcv[1],tcv[0]);
  ang2[0] = ang[0];

  double s1 = sin(ang[0]);
  double c1 = cos(ang[0]);
  t1 = s1 * arm[0][0] - c1 * arm[1][0];
  t2 = s1 * arm[0][1] - c1 * arm[1][1];
  ang[DOF - 1] = atan2(t1,t2);
  ang2[DOF - 1] = ang[DOF - 1];

  for(i = 0;i < 3;i++)
   tcv[i + 3] = -exp(ang[DOF - 1]/PI) * arm[i][2];

  q234 = atan2((c1 * tcv[3] + s1 * tcv[4]),tcv[5]);

  double c234 = cos(q234);
  double s234 = sin(q234);

  b1 = c1 * tcv[0] + s1 * tcv[1] - ak[3] * c234 + dk[DOF - 1]
* s234;
  b2 = dk[0] - ak[3] * s234 - dk[DOF - 1] * c234 - tcv[2];
  modb = b1 * b1 + b2 * b2;
  t1 = modb - ak[1] * ak[1] - ak[2] * ak[2];
  t2 = 2 * ak[1] * ak[2];
  ang[2] = acos(t1 / t2);
  ang2[2] = -acos(t1 / t2);

  double s3 = sin(ang[2]);
  double c3 = cos(ang[2]);

  t1 = (ak[1] + ak[2] * c3) * b2 - b1 * ak[2] * s3;
  t2 = (ak[1] + ak[2] * c3) * b1 + b2 * ak[2] * s3;

  ang[1] = atan2(t1,t2);
  ang2[1] = ang[1];
  ang[3] = q234 - ang[2] - ang[1];
  ang2[3] = q234 - ang2[2] - ang2[1];

  return ang;
}

double * IK(double ak[],double dk[],double tcv[],double
ang2[])
{
  double *ang = new double[DOF];
  double t1,t2,q234,b1,b2,modb;

  ang[0] = atan2(tcv[1],tcv[0]);
  ang2[0] = ang[0];

  double s1 = sin(ang[0]);
  double c1 = cos(ang[0]);
  t1 = sqrt(tcv[3] * tcv[3] + tcv[4] * tcv[4] + tcv[5] * tcv[5]);
  ang[DOF - 1] = PI * log(t1);
  ang2[DOF - 1] = ang[DOF - 1];

  q234 = atan2((c1 * tcv[3] + s1 * tcv[4]),tcv[5]);
  //cout<<"q234 = "<<q234;
  double c234 = cos(q234);
  double s234 = sin(q234);

  b1 = c1 * tcv[0] + s1 * tcv[1] - ak[3] * c234 + dk[DOF - 1]
* s234;
  b2 = dk[0] - ak[3] * s234 - dk[DOF - 1] * c234 - tcv[2];
  modb = b1 * b1 + b2 * b2;
  t1 = modb - ak[1] * ak[1] - ak[2] * ak[2];
  t2 = 2 * ak[1] * ak[2];
  //cout<<"t1 / t2"<<(t1 / t2);
  ang[2] = acos(t1 / t2);
  ang2[2] = -acos(t1 / t2);

  double s3 = sin(ang[2]);
  double c3 = cos(ang[2]);

  t1 = (ak[1] + ak[2] * c3) * b2 - b1 * ak[2] * s3;
  t2 = (ak[1] + ak[2] * c3) * b1 + b2 * ak[2] * s3;

  ang[1] = atan2(t1,t2);
  ang2[1] = ang[1];
  ang[3] = q234 - ang[2] - ang[1];
  ang2[3] = q234 - ang2[2] - ang2[1];

  return ang;
}

Hctm DK(double ak[],double dk[],double alpha[],double
angle[])
{
 int i;
 Hctm rhino[DOF];
 Hctm arm;
 arm.initIdentity();
```

```
  for(i = 0;i < DOF;i++)
  {
    rhino[i].initLctm(angle,dk,ak,alpha,i + 1);
    //rhino[i].printLctm(i + 1,i);
    //if (i == 2) { pause();clrscr(); }
    arm = arm * rhino[i];
  }
  return arm;
}

void BDA(double w1[],double w2[],double ak[],double
dk[],double alpha[],double ep)
{
  double
*ang2=NULL,*w1ang,*w2ang,*wmid,*w12,*w12mid;
  int i;

  w1ang = IK(ak,dk,w1,ang2);//getch();
  w2ang = IK(ak,dk,w2,ang2);//getch();
  wmid = new double[DOF];

  for(i = 0;i < DOF;i++)
   wmid[i] = (w1ang[i] + w2ang[i]) / 2;
  Hctm arm = DK(ak,dk,alpha,wmid);
  double armmat[4][4];
  arm.getvalues(armmat);
  w12 = getTCV(armmat);
  setlinestyle(SOLID_LINE,0,NORM_WIDTH);
  setfillstyle(SOLID_FILL,LIGHTCYAN);
  int x1,y1,x2,y2,x3,y3,x4,y4;
  getxy(w1,x1,y1,320,280);
  getxy(w2,x2,y2,320,280);
  getxy(w12,x3,y3,320,280);
  setcolor(LIGHTCYAN);

  double ax3,ay3,ax1,ay1,ax2,ay2;
  getcabinetxy(w1,ax1,ay1);
  getcabinetxy(w2,ax2,ay2);
  ax3 = (ax1 + ax2) / 2;
  ay3 = (ay1 + ay2) / 2;
  x4 = ceil(ax3 + 320);
  y4 = ceil(280 - ay3);

  fillellipse(x1,y1,3,3);
  fillellipse(x2,y2,3,3);//getch();
  fillellipse(x4,y4,3,3);
  setcolor(RED);
  circle(x1,y1,1);
  circle(x2,y2,1);
  circle(x4,y4,1);
  //setcolor(LIGHTCYAN);
  //circle(x1,y1,2);
  //circle(x2,y2,2);
  //circle(x3,y3,2);

  //getcabinetxy(w12,ax3,ay3);
  double d = sqrt((ax3 - ax1) * (ax3 - ax1) + (ay3 - ay1) * (ay3
- ay1));
  //circle(x4,y4,1);
  gprintf(20,400,"x1 = %4.2f y1 = %4.2f x2 = %4.2f y2 =
%4.2f x3 = %4.2f y3 = %4.2f",ax1,ay1,ax2,ay2,ax3,ay3);
  if (ay2 < ay1) {
         swap(ax1,ax2);
         swap(ay1,ay2);
         //for(i = 0;i < 6;i++)
           //  swap(w1[i],w2[i]);
       }
  double angxy = atan2(ay2 - ay3,ax2 - ax3);
  angxy = angxy * 180 / PI;
  setcolor(LIGHTMAGENTA);
  if (ax1 > 0)  arc(x4,y4,angxy + 180,angxy,ceil(d));
   else arc(x4,y4,180 - angxy,360 - angxy,ceil(d));
  getch();
  setcolor(BLACK);
  gprintf(20,400,"x1 = %4.2f y1 = %4.2f x2 = %4.2f y2 =
%4.2f x3 = %4.2f y3 = %4.2f",ax1,ay1,ax2,ay2,ax3,ay3);

  if (getDistance(w1,w2) > ep)
      {
      for(i = 0;i < 3;i++)
         w12[i] = (w1[i] + w2[i]) / 2;
      BDA(w1,w12,ak,dk,alpha,ep);
      BDA(w12,w2,ak,dk,alpha,ep);
      }

}

void getcabinetxy(double tcv[],double &x,double &y)
{
  //x = 2 * (tcv[0] + tcv[1] * cos(PI/4));
  //y = tcv[2] + tcv[1] * cos(PI/4);
  x = 2 * (tcv[0]);
  y = tcv[2];

}

void getxy(double tcv[],int &x,int &y,int bx,int by)
{
  double a1,b1;
  getcabinetxy(tcv,a1,b1);
  x = bx + (int)ceil(a1);
  y = by - (int)ceil(b1);

}

double getDistance(double w1[],double w2[])
{
    double t = 0;

    for(int i = 0;i < 3;i++)
   t += (w1[i] - w2[i]) * (w1[i] - w2[i]);
    return (sqrt(t));
}
```

```
void swap(double &a,double &b)
{
 double c = a;
 a = b;
 b = c;
}

void printLctm(double mat [][4],int k,int k1)
{
  int y = wherey(),i,j,x;

  gotoxy(10,y + 2);cout<<"T  = ";
  gotoxy(11,y + 1);cout<<k;
  gotoxy(11,y + 3);cout<<k1;
  gotoxy(15,y);cout<<'Ú';
  gotoxy(15,y + 5);cout<<'À';

  for(i = 0;i < 4;i++)
  {
   gotoxy(15,y + i + 1);
   cout<<"³   ";
   for(j = 0;j < 4;j++)
     printf("%- 10.4f   ",mat[i][j]);
   x = wherex();cout<<'³';
  }
  gotoxy(x,y);cout<<'¿';
  gotoxy(x,y + 5);cout<<'Ù';
  gotoxy(1,y + 6);
}

void printvector(double an[],int len,char *m,char sym)
{
  int y = wherey(),i,x;

  gotoxy(20,y + 2);cout<<m<<"  =  ";x = wherex();
  gotoxy(x,y);cout<<'Ú';
  gotoxy(x,y + len + 1);cout<<'À';

  for(i = 0;i < len;i++)
  {
   gotoxy(x,y + i + 1);
   printf("³   %c%-4d =   %- 12.4f   ³",sym,i + 1,an[i]);
  }

  x = wherex() - 1;
  gotoxy(x,y);cout<<'¿';
  gotoxy(x,y + len + 1);cout<<'Ù';
  gotoxy(1,y + len + 2);
}

void line(char c,int len)
{
  for( ;len;len--)
   cout<<c;
}

void pause(char s[])
{
 gotoxy(30,25);
 cout<<s;getch();
}

int gpause(const char *s,int color,int p,int y)
{
  struct textsettingstype old;//In graphics.h

  gettextsettings(&old);
  settextstyle(SMALL_FONT,HORIZ_DIR,4);
  int c = getcolor(),d = textwidth(s);

  setcolor(color);
  outtextxy((getmaxx() - d)/2,y,s);
  settextstyle(old.font,old.direction,old.charsize);setcolor(c);
  if ((color == getbkcolor()) || p != PAUSE)  return 0;
    else    return getch();
```

VII. Conclusion

A simulation of an efficient method of the BDA was demonstrated in this paper using an articulated robot which was designed and fabricated in the college laboratory. A method of computing the straight line motion between two given points in a 3D space using an articulated robot is demonstrated in this paper. Analytical method is also shown here for convenience along with the simulation study. The real time implementation of the same was also carried out using an indigenously developed 5 axis articulated robot in the college. The mathematical results & the experimental results / simulated results show the effectiveness of the developed method [2].

References

[1] Craig J, *Introduction to Robotics* : *Mechanics, Dynamics & Control,* Addison Wessely, USA, 1986.
[2] Robert, J. Schilling, *Fundamentals of Robotics - Analysis and Control*, PHI, New Delhi.
[3] Klafter, Thomas and Negin, *Robotic Engineering*, PHI, New Delhi.
[4] Fu, Gonzalez and Lee, *Robotics: Control*, *Sensing*, *Vision and Intelligence*, McGraw Hill.
[5] Groover, Weiss, Nagel and Odrey, *Industrial Robotics*, McGraw Hill.
[6] Ranky, P. G., C. Y. Ho, *Robot Modeling, Control & Applications*, IFS Publishers, Springer, UK.
[7] Crane, Joseph Duffy, *Kinematic Analysis of Robotic Manipulators*, Cambridge Press, UK.
[8] Manjunath, T.C., (2005), *Fundamentals of Robotics*, Fourth edn., Nandu Publishers, Mumbai.
[9] Manjunath, T.C., (2005), *Fast Track to Robotics*, Second edn., Nandu Publishers, Mumbai.
[10] Dhananjay K Teckedath, *Image Processing*, Third edn., Nandu Publishers, Mumbai.
[11] Gonzalvez and Woods, *Digital Image Processing*, Addison Wesseley Publishers.
[12] Anil K Jain, *Digital Image Processing*, Prentice Hall, Englewood Cliffs, New Jersey, USA.
[13] http://www.wikipedia.org
[14] Michael Dipperstein, *Run Length Encoding* (*RLE*) *Discussion and Implementation*.
[15] Flusser, J.; Suk, T.; Saic, S., *Recognition of blurred images by the method of moments*, Image Processing, IEEE Transactions.

[16] Bob Bailey, *Moments in Image Processing*, Nov. 2002.
[17] Phillip Coiffette, (1995), Robotics Series, Volume I to VIII, Kogan Page, London, UK.
[18] William Burns and Janet Evans, (2000), Practical Robotics - Systems, Interfacing, Applications, Reston Publishing Co.
[19] Yoshikawa T., (1984), "Analysis and Control of Robot Manipulators with Redundancy", Proc. First Int. Symp. on Robotics Research, Cambridge, MIT Press, pp. 735-748.
[20] Whitney DE., (1972), "The Mathematics of Coordinated Control of Prosthetic Arms and Manipulators", Trans. ASM J. Dynamic Systems, Measurements and Control, Vol. 122, pp. 303-309.
[21] Lovass Nagy V, R.J. Schilling, (1987), "Control of Kinematically Redundant Robots Using {1}-inverses", *IEEE Trans. Syst. Man, Cybernetics*, Vol. SMC-17 (No. 4), pp. 644-649.
[22] Lovass Nagy V., R J Miller and D L Powers, (1978), "An Introduction to the Application of the Simplest Matrix-Generalized Inverse in Systems Science", *IEEE Trans. Circuits and Systems*, Vol. CAS-25 (No. 9), pp. 776.

T.C.Manjunath, born in Bangalore, Karnataka, India on Feb. 6, 1967 received the B.E. Degree in Electrical Engineering from the University of Bangalore in 1989 in First Class and M.E. in Electrical Engineering with specialization in Automation, Control and Robotics from the University of Gujarat in 1995 in First Class with Distinction and Ph.D. from the Interdisciplinary Programme in Systems and Control Engineering Department of Indian Institute of Technology Bombay in the year 2007, respectively. He has got a teaching experience of nearly 20 long years in various engineering colleges all over the country and is currently working as Professor and Head of the Department of Electronics and Communication Engineering in New Horizon College of Engineering in Bangalore, Karnataka, India. He also worked as a Research Engineer in the Systems and Control Engineering (IIT Bombay, India) and worked on control of space launch vehicles using FOS feedback technique. He has published a number of papers in the various National, International journals and Conferences and published two textbooks on Robotics. He also published a research monograph in the International level from the Springer-Verlag publishers based on his Ph.D. thesis topic titled, "Modeling, Control and Implementation of Smart Structures", Vol. 350, LNCIS, costing 79.95 Euros. He was a student member of IEEE for 6 years, SPIE student member and IOP student member for 4 years, life member of ISSS (India), life member of the ISTE (India), life member of ISOI (India), life member of SSI (India) and life member of the CSI (India) and a fellow of the IETE (India). He has visited Singapore, Russia, United States of America and Australia for the presentation of his research papers in various international conferences. His biography was published in 23rd edition of Marquis's Who's Who in the World in the 2006 issue. He has also guided more than 2 dozen undergraduate and post-graduate projects. Many of his guided projects, interviews have appeared in various national newspapers and magazines. He has also presented a number of guest lectures and various seminars and participated in more than a dozen CEP / DEP courses, seminars, workshops, symposiums in the various parts of the country in different institutions and also conducted a few courses. His current research interests are in the area of Robotics, Smart Structures, Control systems, Network theory, Mechatronics, Process Control and Instrumentation, Electromagnetic fields and waves, MATLAB, Signals and systems (CT and DT), Industrial automation, Artificial intelligence, Digital signal processing, Digital Image Processing, Periodic output feedback control, Fast output feedback control, Sliding mode control of SISO and multivariable systems and many of the control related subjects and its allied labs and their various applications.