References

[1]. Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng, *Learning Depth from Single Monocular Images* [Online], Available: http://ai.stanford.edu/~asaxena/learningdepth/NIPS_LearningDepth.pdf

[2]. Ashutosh Saxena, Min Sun and Andrew Y. Ng, *Make3D: Depth Perception from a Single Still Image* [Online], Available: http://robotics.stanford.edu/~ang/papers/aaai08-Make3dDepthPerceptionSingleImage.pdf

[3]. Francisco Bonin-Font, Alberto Ortiz, Gabriel Oliver, *Visual Navigation for Mobile Robots: a Survey*, [Online], Available: http://dmi.uib.es/~fbonin/survey.pdf

[4]. Yoko Watanabe, *Stochastically Optimized Monocular Vision-Based Navigation And Guidance* [Online], Available: http://smartech.gatech.edu/jspui/bitstream/1853/22545/1/watanabe_yoko_200804_phd.pdf

[5]. Animesh Garg, Anju Toor, Sahil Thakkar, Shiwangi Goel, Sachin Maheshwari, Satish Chand, *Object Identification and Mapping using Monocular Vision in an Autonomous Urban Driving System* [Online], Available: http://www.ijcte.org/icmv/ICMV2010/136-ICMV2010-W12016.pdf

[6]. Tobias Low, Antoine Manzanera, *Ground-Plane Classification for Robot Navigation* [Online], Available: http://www.ensta-paristech.fr/~manzaner/Publis/icarcv10.pdf

[7]. Nicolau Leal Werneck, Anna Helena Reali Costa, *Mapping with Monocular Vision in Two Dimensions* [Online], Available: http://nwerneck.sdf.org/almoxarifado/WerneckCosta10.pdf

[8]. Chau Nguyen Viet, Ian Marshall, *VISION-BASED OBSTACLE AVOIDANCE FOR A SMALL, LOW-COST ROBOT* [Online], Available: http://eprints.lancs.ac.uk/27256/1/27256.pdf

[9]. Jiandong Tian, Yandong Tang, *Learning and Vision-Based Obstacle Avoidance and Navigation* [Online], Available: http://ir.sia.ac.cn/bitstream/173321/9008/2/ZZZJ000014.pdf

[10]. Jan Hoffmann, Matthias J¨ungel, Martin L¨otzsch, *A Vision Based System for Goal-Directed Obstacle Avoidance* [Online], Available: http://www2.informatik.hu-berlin.de/~juengel/papers/hoffmann-juengel-loetzsch-rc04.pdf

[11]. Jan Hoffmann, Matthias J¨ungel, Martin L¨otzsch, *A Vision Based System for Goal-Directed Obstacle Avoidance used in the RC'03 Obstacle Avoidance Challenge* [Online], Available: http://martin-loetzsch.de/publications/rc04-obstacle.pdf

[12]. Mehmet Serdar Guzel, Robert Bicker, *Vision Based Obstacle Avoidance Techniques* [Online], Available: http://cdn.intechopen.com/pdfs/24918/InTech-Vision_based_obstacle_avoidance_techniques.pdf

[13]. Daniel Maier, Maren Bennewitz, Cyrill Stachniss, *Self-supervised Obstacle Detection for Humanoid Navigation Using Monocular Vision and Sparse Laser Data* [Online], Available: http://hrl.informatik.uni-freiburg.de/papers/maier11icra.pdf

[14]. Sean Quinn Marlow, Jack W. Langelaan, *Local Terrain Mapping for Obstacle Avoidance using Monocular Vision* [Online], Available: http://www.aero.psu.edu/avia/pubs/MarLan09a.pdf

[15]. Scott Lenser, Manuela Veloso, *Visual Sonar: Fast Obstacle Avoidance Using Monocular Vision* [Online], Available: http://www.cs.cmu.edu/~robosoccer/cmrobobits/papers/03iros-sonar.pdf

[16]. Jef Michels, Ashutosh Saxena, Andrew Y. Ng, *High Speed Obstacle Avoidance using Monocular Vision and Reinforcement Learning* [Online], Available: http://www.cs.cornell.edu/~asaxena/rccar/icml_obstacleavoidance.pdf

[17]. Juan Fasola, Paul E. Rybski, Manuela M. Veloso, *Fast Goal Navigation With Obstacle Avoidance Using A Dynamic Local Visual Model* [Online], Available: http://www.cs.cmu.edu/~mmv/papers/05sbai-juan.pdf

[18]. Ashutosh Saxena, *Monocular Depth Perception And Robotic Grasping Of Novel Objects* [Online], Available: http://www.irisa.fr/lagadic/pdf/2011_civts_cherubini.pdf

[19]. Iwan Ulrich, Illah Nourbakhsh, *Appearance-Based Obstacle Detection with Monocular Color Vision* [Online], Available: http://www.cs.cmu.edu/~illah/PAPERS/abod.pdf

[20]. S. Wybo, D. Tsishkou, C. Vestri, F. Abad, R. Bendahan, and S. Bougnoux, *Obstacles Avoidance By Monocular Multi-Cue Image Analysis* [Online], Available: http://vestri.free.fr/data/ITS2008.pdf

[21]. Ian Lenz, Mevlana Gemici, Ashutosh Saxena, *Low-Power Parallel Algorithms for Single Image based Obstacle Avoidance in Aerial Robots* [Online], Available: http://www.cs.cornell.edu/~asaxena/papers/lowpower_obstacleavoidance_mav.pdf

[22]. G.C.H.E. de Croon, B.D.W. Remes, C. De Wagter, R. Ruijsink, *Sky Segmentation Approach to Obstacle Avoidance* [Online], Available: http://www.delfly.nl/Sky%20Segmentation%20Approach%20to%20Obstacle%20Avoidance(Draft).pdf

[23]. Ashutosh Saxena, Sung H. Chung, Andrew Y. Ng, *Learning Depth from Single Monocular Images* [Online], Available: http://www.cs.cmu.edu/~efros/courses/AP06/Papers/saxena-nips-05.pdf

[24]. Andrea Cherubini, Fabien Spindler, Franc¸ois Chaumette, *A Redundancy-Based Approach For Visual Navigation With Collision Avoidance* [Online], Available: http://www.cs.cornell.edu/~asaxena/thesis_saxena.pdf

[25]. Dzung L. Pham, *Spatial Models for Fuzzy Clustering* [Online], Available: http://www.idealibrary.com/10.1.1.89.9945.pdf

[26]. Koushik Mondal, Paramartha Dutta, Siddhartha Bhattacharyya, *Gray Image extraction using Fuzzy Logic* [Online], Available: http://arxiv.org/ftp/arxiv/papers/1206/1206.4391.pdf

[27]. P. Sobrevilla, E. Montseny, *Fuzzy Sets in Computer Vision: an Overview* [Online], Available: http://upcommons.upc.edu/revistes/bitstream/2099/1735/1/article.pdf

[28]. Rafaelmun˜ Oz-Salinas, Eugenio Aguirre, Miguel Garcı´A-Silvente, Antonio Gonza´ Lez, *Door-detection using computer vision and fuzzy logic* [Online], Available: http://www.wseas.us/e-library/conferences/athens2004-b/papers/474-224.pdf

[29]. Cristina M. Peixoto Santos, Manuel Jo˜ao Ferreira, *Control of an Industrial Desktop Robot Using Computer Vision and Fuzzy Rules* [Online], Available: http://repositorium.sdum.uminho.pt/bitstream/1822/4328/1/D7-02.pdf

[30]. Cristina P Santos, Manuel J Ferreira, *Computer Vision and Fuzzy Rules Applied to a Dispensing Application in an Industrial Desktop Robot* [Online], Available: http://cdn.intechopen.com/pdfs/274/InTech-Computer_vision_and_fuzzy_rules_applied_to_a_dispensing_application_in_an_industrial_desktop_robot.pdf

[31]. James M. Keller, Pascal Matsakis, *Aspects of High Level Computer Vision Using Fuzzy Sets* [Online], Available: http://www.cis.uoguelph.ca/~matsakis/Publications/FUZZ99.pdf

[32]. Dimitris K. Iakovidis, Nikos Pelekis, Evangelos Kotsifakos, Ioannis Kopanakis, *Intuitionistic Fuzzy Clustering with Applications in Computer Vision* [Online], Available: http://www.e-bilab.com/Portals/0/Papers/Papers/P18%20Intuitionistic%20Fuzzy%20Clustering%20with%20Applications%20in%20Computer%20Vision/Intuitionistic%20Fuzzy%20Clustering%20with%20Applications%20in%20Computer%20Vision.pdf

[33]. Kluwer Academic Publishers, *Editorial:Neural-Fuzzy Applications in Computer Vision* [Online], Available: http://users.cs.cf.ac.uk/Paul.Rosin/resources/papers/JIRS.pdf

[34]. İsmail H. Altaş, Adel M Sharaf, *A Photovoltaic Powered Tracking System for Moving Objects* [Online], Available: http://www.emo.org.tr/ekler/99c61acedb54c52_ek.pdf

Appendix A:

Improvements on Image Processing and Optical Flow Algorithms - Code Level Implementation

A.1. Introduction

This appendix includes the code-level implementations of improvements done for the existing Optical Flow algorithms to detect a moving object. These improvements consist of enhancement of overall efficiency i.e. using pointers - direct memory addresses, noise filtering with statistical analytical techniques, marking the region of interest and edge detection for isolating the object in interest.

A.2. Structure of the Code -An Abstract View

Below (Code Fragment A.1) is the code written in C++ programming language - implementation of OpenCV LK-Optical Flow based object tracking with innovative improvements. Implementation of key methods will be under the relevant sub-section.


```
#include "stdafx.h"
#include "opencv2/video/tracking.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <iostream>
#include <ctype.h>

using namespace cv;
using namespace std;

Point2f pt;
bool addRemovePt = false;

Mat test;
unsigned short qaudrant[4]={0,0,0,0};
unsigned short qaudrnt,qaudrnt2;
vector<uchar> statusRfnd;
unsigned int leftX,rightX,topY,bottomY;
int meanX,meanY;

cv::Mat croppedImage;

IplImage* base;

//Statistical Approach I - Detect dominating quadrant (Noise Removal)
static void detectDominentQuadrant(Mat& frame, const vector<Point2f>& prevPts, const
vector<Point2f>& nextPts, const vector<uchar>& status)
{
	//See section A.3. Statistical Approach I for implementation
}
```

```
//Statistical Approach I - Draw arrows to visualize actual predicted motion vectors
static void drawArrowsRefined(Mat& frame, const vector<Point2f>& prevPts, const
vector<Point2f>& nextPts, const vector<uchar>& status, unsigned short quadrant,
unsigned short quadrant2, bool enableQ2, Scalar line_color = Scalar(0, 255, 0))
{
	//See section A.3. Statistical Approach I for implementation
}

//Statistical Approach II to Mark ROI (Region of Interest)
void defineROI(Mat& frame, const vector<Point2f>& prevPts, const vector<uchar>&
statusRfnd, int minCount)
{
	//See section A.4. Statistical Approach II to Mark ROI (Region of Interest) for
	implementation
}

//Canny Edge Detection (An auxiliary type conversion)
Mat CannyEdgeMat(Mat img)
{
	//See section A.5. Edge Detection to Image Segmentation for implementation
}

//Canny Edge Detection to separate object in interest
void edgesWithinROI(Mat& frame, Mat edgImg, const vector<Point2f>& prevPts,
vector<uchar>& statusRfnd)
{
	//See section A.5. Edge Detection to Image Segmentation for implementation
}

int main( int argc, char** argv )
{
	VideoCapture cap;
	TermCriteria termcrit(CV_TERMCRIT_ITER|CV_TERMCRIT_EPS,20,0.03);
	Size winSize(10,10);
	//Size winSize(15,15);

	const int MAX_COUNT = 1000;
	bool needToInit = false;
	bool nightMode = false;

	if( argc == 1 || (argc == 2 && strlen(argv[1]) == 1 && isdigit(argv[1][0])))
		cap.open(argc == 2 ? argv[1][0] - '0' : 0);
	else if( argc == 2 )
		cap.open(argv[1]);

	if( !cap.isOpened() )
	{
		cout << "Could not initialize capturing...\n";
		return 0;
	}

	help();

	namedWindow( "(C)2013 Motion Detect - AI Lab, UoM", 1 );

	setMouseCallback("Edges Detected", onMouse, 0 );

	Mat gray, prevGray, image;
	vector<Point2f> points[2];
```

```
for(;;)
	{
		Mat frame;
		cap >> frame;
		if( frame.empty() )
			break;

		frame.copyTo(image);
		cvtColor(image, gray, CV_BGR2GRAY);

		if( nightMode )
			image = Scalar::all(0);

		if( needToInit )
		{
			//.........Automatic initialization (MOTION VECTORS
			DETECTION).............
			/*goodFeaturesToTrack(InputArray image, OutputArray corners,
			int maxCorners, double qualityLevel, double minDistance,
			InputArray mask=noArray(), int blockSize=3,
			bool useHarrisDetector=false, double k=0.04 )*/

			goodFeaturesToTrack(gray, points[1], MAX_COUNT, 0.01, 5, Mat(),
			3, 0, 0.04);
			cornerSubPix(gray, points[1], winSize, Size(-1,-1), termcrit);
		}
		else if( !points[0].empty() )
		{
			vector<uchar> status;
			vector<float> err;
			if(prevGray.empty())
				gray.copyTo(prevGray);
			calcOpticalFlowPyrLK(prevGray, gray, points[0], points[1],
			status, err, winSize,3, termcrit, 0);

			detectDominentQuadrant(image,points[0],points[1],status);

			drawArrowsRefined(image,points[0],points[1],
			status,qaudrnt,qaudrnt2,true);
			defineROI(image, points[0], statusRfnd, 20);

			edgesWithinROI(image, test, points[0], statusRfnd);
		}

		needToInit = false;
		imshow("(C)2013 Motion Detect - AI Lab, UoM", image);

		char c = (char)waitKey(10);
		if( c == 27 )
			break;
		switch( c )
		{
		case 'r':
			needToInit = true;
			break;
		case 'c':
			points[1].clear();
			break;
		default:
			;
		}
```

```
                std::swap(points[1], points[0]);
                swap(prevGray, gray);
        }

        return 0;
}
```

Code Fragment A.1: Implementation of Image Processing Module (in abstract view)

A.3. Statistical Approach I

Below (Code Fragment A.2) is the source code for implementation of Statistical Approach I.

```
//Statistical Approach I - Detect dominating quadrant (Noise Removal)
static void detectDominentQuadrant(Mat& frame, const vector<Point2f>& prevPts, const
vector<Point2f>& nextPts, const vector<uchar>& status)
{
        unsigned short temp,max;
        for(int i=0;i<4;i++)
        {
                qaudrant[i]=0;
        }

        for (size_t i = 0; i < prevPts.size(); ++i)
        {
                if (status[i])
                {
                        int line_thickness = 1;

                        Point p = prevPts[i];
                        Point q = nextPts[i];

                        if(q.y>p.y)
                        {
                                if(q.x>p.x) // Point belongs to 1st quadrant
                                {
                                        temp=qaudrant[0];
                                        temp++;
                                        qaudrant[0]=temp;
                                }
                                if(q.x<p.x)// Point belongs to 2nd quadrant
                                {
                                        temp=qaudrant[1];
                                        temp++;
                                        qaudrant[1]=temp;
                                }
                        }
                        if(q.y<p.y)//y<0
                        {
                                if(q.x<p.x)//y<0, x<0 Point belongs to 3rd quadrant
                                {
                                        temp=qaudrant[2];
                                        temp++;
                                        qaudrant[2]=temp;
                                }
                                if(q.x>p.x)// y<0, x>0 Point belongs to 4th quadrant
                                {
                                        temp=qaudrant[3];
                                        temp++;
```

```
                                qaudrant[3]=temp;
                        }
                }
        }
    }//end of for

    //Find the dominating quadrant
    unsigned short tmp;
    max=qaudrant[0];
    qaudrnt=1;
    for(int i=1;i<4;i++)
    {
        tmp=qaudrant[i];
        if(tmp>max)
        {
            qaudrnt=i+1;
        }
    }
    //Find the 2nd dominating quadrant
    qaudrant[qaudrnt-1]=0; // Replace the maximum (ready for next highest)
    max=qaudrant[0];
    qaudrnt2=1;
    for(int i=1;i<4;i++)
    {
        tmp=qaudrant[i];
        if(tmp>max)
        {
            qaudrnt2=i+1;
        }
    }
}



//Statistical Approach I - Draw arrows to visualize actual predicted motion vectors
static void drawArrowsRefined(Mat& frame, const vector<Point2f>& prevPts, const
vector<Point2f>& nextPts, const vector<uchar>& status, unsigned short quadrant,
unsigned short quadrant2, bool enableQ2, Scalar line_color = Scalar(0, 255, 0))
{
    bool isValid=false;
    unsigned short minVel=0;
    unsigned short altV=1;
    unsigned short altH=1;

    statusRfnd=status;
    int sumX=0;
    int sumY=0;
    int count=0;

    for (size_t i = 0; i < prevPts.size(); ++i)
    {
        if (status[i])
        {
            int line_thickness = 1;

            Point p = prevPts[i];
            Point q = nextPts[i];

            if(q.y>p.y && q.y-p.y>=minVel)
            {
                if(q.x>p.x && q.x-p.x>=minVel) // Point belongs to 1st
                quadrant
                {
                    if(quadrant==1)
```

```
                        isValid=true;
                    if(enableQ2==true && quadrant2==1)
                        isValid=true;
                }
                if(q.x<p.x && p.x-q.x>=minVel)// Point belongs to 2nd
                quadrant
                {
                    if(quadrant==2)
                        isValid=true;
                    if(enableQ2==true && quadrant2==2)
                        isValid=true;
                }
            }
            if(q.y<p.y && p.y-q.y>=minVel)//y<0
            {
                if(q.x<p.x && p.x-q.x>=minVel)//y<0, x<0 Point belongs to
                3rd quadrant
                {
                    if(quadrant==3)
                        isValid=true;
                    if(enableQ2==true && quadrant2==3)
                        isValid=true;
                }
                if(q.x>p.x && q.x-p.x>=minVel)// y<0, x>0 Point belongs to
                4th quadrant
                {
                    if(quadrant==4)
                        isValid=true;
                    if(enableQ2==true && quadrant2==4)
                        isValid=true;
                }
            }
            if(abs(q.y-p.y)<altV)
                isValid=false;
            if(abs(q.x-p.x)<altH)
                isValid=false;

            if(isValid)
            {
                count++;
                sumX+=p.x;
                sumY+=p.y;

                double angle = atan2((double) p.y - q.y, (double) p.x -
                q.x);

                double hypotenuse = sqrt( (double)(p.y - q.y)*(p.y - q.y)
                + (double)(p.x - q.x)*(p.x - q.x) );

                if (hypotenuse < 1.0)
                    continue;

                //Lengthen the arrow by a factor of three.
                q.x = (int) (p.x - 3 * hypotenuse * cos(angle));
                q.y = (int) (p.y - 3 * hypotenuse * sin(angle));

                //draw the main line of the arrow.
                line(frame, p, q, line_color, line_thickness);

                //draw the tips of the arrow. do some scaling so that the
                // tips look proportional to the main line of the arrow.
```

```
                        p.x = (int) (q.x + 9 * cos(angle + CV_PI / 4));
                        p.y = (int) (q.y + 9 * sin(angle + CV_PI / 4));
                        line(frame, p, q, line_color, line_thickness);

                        p.x = (int) (q.x + 9 * cos(angle - CV_PI / 4));
                        p.y = (int) (q.y + 9 * sin(angle - CV_PI / 4));
                        line(frame, p, q, line_color, line_thickness);

                }
                else //isValid=false
                        statusRfnd[i]=false;
        }
    }//end of for

    if(count!=0)
    {
            meanX=sumX/count;
            meanY=sumY/count;
    }
}
```

Code Fragment A.2: implementation of Statistical Approach I

A.4. Statistical Approach II to Mark ROI (Region of Interest)

Below (Code Fragment A.3) is the source code for implementation of Statistical Approach II for marking ROI (Region of Interest).


```
//Statistical Approach II to Mark ROI (Region of Interest)
void defineROI(Mat& frame, const vector<Point2f>& prevPts, const vector<uchar>&
statusRfnd, int minCount)
{
    double eucladian;
    double sumEucladian=0;
    int sumN=0;
    int radious=0;

    for (size_t i = 0; i < prevPts.size(); ++i)
    {
            if (statusRfnd[i])
            {
                    sumN++;
                    Point p = prevPts[i];
                    eucladian=sqrt( (double)(p.y - meanY)*(p.y - meanY) +
                    (double)(p.x - meanX)*(p.x - meanX) );
                    sumEucladian+=eucladian;
            }
    }//end of for
    if(sumN>=minCount)
    {
            if(sumN!=0)
                    radious=(int)(sumEucladian/sumN);
            Point p(meanX,meanY);
            circle( frame, p, 3, Scalar(0,255,255), -1, 8);
            circle( frame, p, radious, Scalar(0,255,255), 1, 8);
    }

    leftX=1024;
    topY=768;
    rightX=0;
```

```
	bottomY=0;

	for (size_t i = 0; i < prevPts.size(); ++i)
	{
		if (statusRfnd[i])
		{
			Point p = prevPts[i];
			eucladian=sqrt( (double)(p.y - meanY)*(p.y - meanY) +
			(double)(p.x - meanX)*(p.x - meanX) );
			if(eucladian<=radious) // Point within ROI
			{
				if(p.x<leftX)
					leftX=p.x;
				if(p.x>rightX)
					rightX=p.x;
				if(p.y>bottomY)
					bottomY=p.y;
				if(p.y<topY)
					topY=p.y;
			}
			else
				statusRfnd[i]=false; // Remove point
		}
	}//end of for
	//Mark the determined ROI
	rectangle(frame,cvPoint(leftX,topY), cvPoint(rightX, bottomY),cvScalar(0, 255,
	0),2,8,0);
}
```

Code Fragment A.3: Implementation of Statistical Approach II

A.5. Edge Detection to Image Segmentation

Below (Code Fragment A.4) is the source code for implementation of Canny Edge Detection to separate the object in interest, which is obviously inclusive in ROI marked in previous stage.

```
//Canny Edge Detection
Mat CannyEdgeMat(Mat img)
{
	//a type conversion is required beforehand
	//Convert Mat to IplImage*
	IplImage* cvGetImage( const CvArr* arr, IplImage* image_header );
	IplImage* base = &img.operator IplImage();

	//Apply Canny Edge Detection
	cvCanny( base, base, 10, 100, 3 );

	//Convert IplImage* to Mat
	Mat imgMat(base);
	Mat Mimg=base;

	return Mimg;
	cvReleaseImage( &base );
}

//Canny Edge Detection within cropped area -ROI : to separate object in interest
void edgesWithinROI(Mat& frame, Mat edgImg, const vector<Point2f>& prevPts,
vector<uchar>& statusRfnd)
{
```

```
	for(int i=0; i<edgImg.cols; i++)
	{
		for(int j=0; j<edgImg.rows; j++)
		{
			try
			{
			if((int)edgImg.at<uchar>(j,i)==255)//edge (white pixel)
			{
				Point p=(j,i);
				circle( frame, p, 1, Scalar(0,0,255), -1, 8);
			}
			}
			catch(...)
			{

			}
		}
	}
}
```

Code Fragment A.4: implementation of Canny Edge Detection

Appendix B:

Real-World Data Collected for Constructing the FMM

B.1. Introduction

This appendix includes the real world data captured on experimental basis in order to find the actual relationships (mapping the hypothetical figures) among the parameters relevant to the Fuzzy Mathematical Model (FMM).

B.2. Apparent Size - Distance Variation

Table B.1 denotes the actual relationship between the distance to the object in reality and the apparent size recorded by the camera which used to obtain the primary input to the system. Using these real world data, the mathematical model in Fig.B.1 was fitted, and that is the model implemented in the Motion Path Generator, a Simulator, which produces synthetic data to be used as inputs to the FMM.

Actual Distance (cm)	Apparent Size (pixels)	Actual Distance (cm)	Apparent Size (pixels)
300	31	160	66
295	31	155	68
290	32	150	70
285	35	145	74
280	35	140	76
275	35	135	79
270	37	130	84
265	38	125	87
260	38	120	90
255	39	115	95
250	40	110	99
245	41	105	106
240	42	100	109
235	43	95	117
230	44	90	123
225	45	85	129
220	47	80	139
215	47	75	148
210	47	70	162
205	51	65	175
200	51	60	190
195	54	55	206
190	55	50	231
185	56	45	256
180	58	40	290
175	60	35	339
170	63	30	399
165	64		

Table B.1: Relationship between the distance to the object and the apparent size

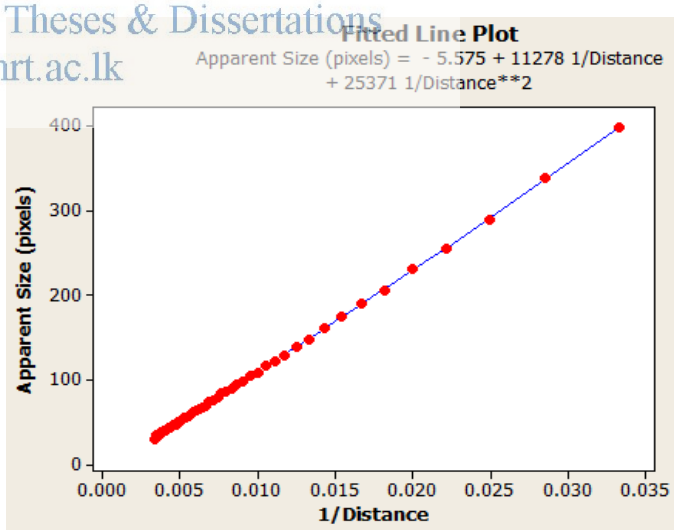

Fig.B.1: Mathematical model for relationship between depth and apparent size

B.3. Relationship between the Skewness and Gradient (m)

The relationship between the Skewness and gradient (m) of the motion path equation at different intersects(c values) was very useful to determine the primary important information - gradient maintains a relationship with the skewness of the apparent size variation curve, that is independent from the intersect(c) of the motion path equation. Table B.2 verifies the validity of the statement, above mentioned, in details. These data were collected using the simulator named 'Motion Path Generator' which relies on the mathematical model constructed in above B.2. Although these data are synthetic, it was verified that the simulator functions properly and generates precise data as same as real world, repeated and confirmed. Therefore these can be considered as virtual real data, very similar to actual experience in experimental environments. See evaluation of the simulator for more details.

m	*c*	*Sk*	*m*	*c*	*Sk*
-0.104	605	0.9893	-0.598	450	-0.0011
-0.104	547	0.9893	-0.602	359	-0.0019
-0.104	485	0.9889	-0.6	267	0.0085
-0.104	426	0.9864	-0.603	176	-0.0084
-0.105	364	0.9866	-0.583	87	0.0047
-0.106	303	0.9883	-0.703	414	-0.1485
-0.107	242	0.9814	-0.701	333	-0.1347
-0.108	181	0.9788	-0.702	239	-0.1431
-0.109	122	0.9555	-0.697	163	-0.1471
-0.115	59	0.922	-0.694	78	-0.1472
-0.201	574	0.8291	-0.797	384	-0.2686
-0.199	460	0.8342	-0.797	310	-0.2601
-0.203	345	0.8252	-0.798	229	-0.2707
-0.2	232	0.8318	-0.794	151	-0.2637
-0.205	114	0.8123	-0.794	74	-0.2588
-0.299	544	0.5452	-0.899	349	-0.4003
-0.3	435	0.5559	-0.898	280	-0.3998
-0.298	326	0.5517	-0.899	208	-0.404
-0.299	217	0.5728	-0.902	137	-0.3947
-0.293	108	0.5591	-0.906	66	-0.4108
-0.5	481	0.1535	-1.007	318	-0.5213
-0.496	384	0.156	-1	256	-0.5202
-0.5	288	0.1516	-1.011	188	-0.5255
-0.5	190	0.1518	-1	121	-0.5499
-0.5	93	0.1555	-1	57	-0.5598
-0.591	451	0.0037			

Table B.2: Skewness and gradient (m) of the motion path at different intersects(c values)

B.4. Relationship between the Intersect (c), Initial Apparent Size and Gradient (m)

This relationship is bit complicated compared to gradient-skewness relationship. The strategy followed to realize that was as follows:

The variation of the intersect (c) and the initial apparent size of the object was studied while maintaining the gradient (m) uniform at a particular instance. Then again m was set to a different value, kept constant and repeated the experiment. Table B.3. shows the experimental results. Fig.B.2 illustrates a summarized form of observations.

m ≈ - 0.1	c	initial size
-0.104	606	10
-0.099	577	11
-0.104	546	12
-0.098	516	13
-0.104	485	14
-0.097	456	15
-0.098	442	16
-0.101	427	17
-0.097	397	18
-0.103	379	19
-0.098	365	20
-0.103	348	21
-0.096	335	22
-0.102	328	23
-0.099	317	24
-0.096	304	25
-0.101	292	26
-0.102	284	27
-0.096	274	28
-0.098	269	29
-0.101	259	30
-0.103	255	31
-0.105	248	32
-0.096	243	33
-0.099	233	34
-0.1	231	35
-0.105	220	36
-0.105	218	37
-0.097	210	38
-0.097	206	39
-0.099	201	40
-0.101	198	41
-0.104	193	42
-0.106	185	43
-0.092	184	44
-0.094	179	45
-0.095	174	47
-0.098	169	48
-0.102	163	50
-0.105	160	51
-0.109	155	53
-0.091	151	54
-0.094	148	55
-0.096	144	57
-0.098	139	59
-0.102	136	60
-0.104	131	62
-0.109	127	64
-0.109	124	65
-0.091	121	68
-0.091	118	69
-0.093	116	71
-0.095	112	73
-0.1	111	74
-0.103	105	77
-0.103	102	79
-0.103	101	80

m ≈ - 1.0	c	initial size
-1.007	317	10
-1.008	295	11
-1	285	12
-1.008	267	13
-0.991	257	14
-0.991	246	15
-1	228	16
-1	219	17
-1.011	204	18
-1.011	194	19
-1	193	20
-1	186	21
-1	170	22
-1	166	23
-1	159	24
-1	154	25
-1	148	26
-1	145	27
-1	138	28
-1	136	29
-1	133	30
-1	126	31
-1	123	32
-1	118	33
-1	115	34
-1	115	35
-1	112	36
-1	110	37
-1	104	38
-1	101	40
-1	96	41
-1	93	43
-1	88	45
-1	85	46
-1	82	48
-1	78	50
-1	75	53
-1	72	54
-1	71	55
-1	71	56
-1	68	58
-1	64	61
-1	61	63
-1	59	65
-1	54	67
-1	55	69
-1	51	72
-1	50	75
-1	49	77
-1	48	78
-1	47	80
-1	44	81
-1	42	84
-1	43	86
-1	41	88
-1	39	91
m ≈ - 2.0	**c**	**initial size**

-0.108	97	83
-0.114	95	85
-0.118	89	90

-1.9991	4	10
-2	-16	51
-1.938	-9	85

Table B.3. Relationship between the Intersect (c), Initial Apparent Size and Gradient (m)

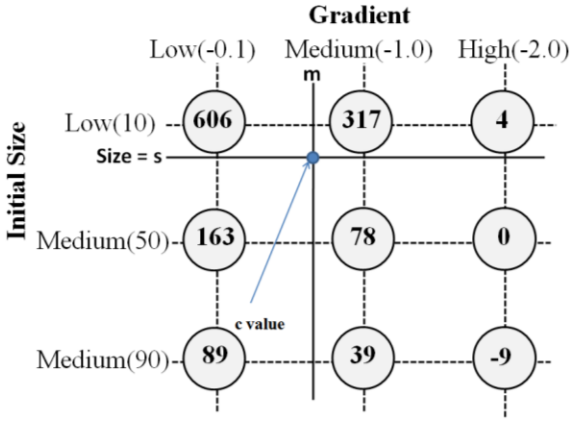

Fig.B.2: Variation of c with m and initial size

B.5. Real World Data to Realize Fuzzy Membership Functions

Fuzzy membership functions are maintained as value matrices or look-up tables, more specifically as 2-D arrays in point of view of implementation. Section B.5 includes the experimental data those were used as the basis of each membership function.

B.5.1. Fuzzy Membership of *m* and *Sk* ($0 \leq SK \leq 1.0$)

The objective of collecting real world data included in Table B.4 was filling the top row of the look-up table (array in point of view of implementation) denoted by Fig. B.3.

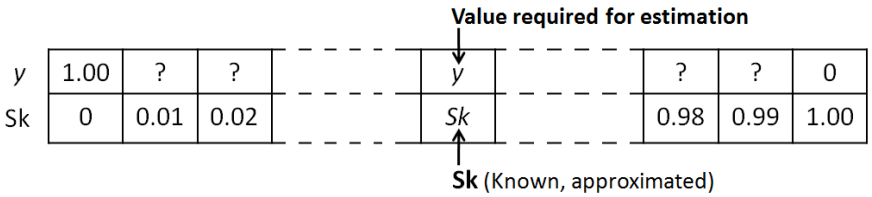

Fig. B.3: Look-up table for *Sk* and fuzzy membership values

The strategy followed was, substituting the *m* value reported at corresponding *Sk* to the equation Eq(1) stated below. In order to succeed that, a large no. of motion paths were generated by the simulator, then Sk values of the apparent size variation curve relevant to the particular simulated motions were recorded. Then the value pairs were sorted with ascending order, order by Sk. The next step was deriving a new table, the first two columns of Table B.4. The precision of the Motion Path Generator's *m* coefficient was upgraded up to three decimal places beforehand. And note that m values for correspond Sks those appear in Table B.4. with a precision of four decimal places, that is due to having the average of several *m* values mapping to the same Sk (approximately) in several experimental attempts. Fig.B.4. is the ultimate outcome.

$$m = [y*(-0.6)] + [(1-y)*(-0.1)]$$

$$\Rightarrow \mathbf{y = (m + 0.1) / (-0.5)} \longleftarrow \mathbf{Eq(1)}$$

Sk	*m*	fuzzy(*y*)	fuzzy(*y*-1)	Sk	*m*	fuzzy(*y*)	fuzzy(*y*-1)
0	-0.5979	0.9958	0.0042	0.51	-0.3204	0.4408	0.5592
0.01	-0.59	0.9801	0.0199	0.52	-0.3164	0.4329	0.5671
0.02	-0.5823	0.9646	0.0354	0.53	-0.3125	0.425	0.575
0.03	-0.5747	0.9494	0.0506	0.54	-0.3086	0.4171	0.5829
0.04	-0.5672	0.9345	0.0655	0.55	-0.3046	0.4093	0.5907
0.05	-0.5599	0.9198	0.0802	0.56	-0.3007	0.4015	0.5985
0.06	-0.5527	0.9054	0.0946	0.57	-0.2968	0.3936	0.6064
0.07	-0.5456	0.8912	0.1088	0.58	-0.2929	0.3858	0.6142
0.08	-0.5386	0.8772	0.1228	0.59	-0.289	0.378	0.622
0.09	-0.5318	0.8635	0.1365	0.6	-0.2851	0.3702	0.6298
0.1	-0.525	0.8501	0.1499	0.61	-0.2812	0.3624	0.6376
0.11	-0.5184	0.8368	0.1632	0.62	-0.2773	0.3546	0.6454
0.12	-0.5119	0.8238	0.1762	0.63	-0.2734	0.3467	0.6533
0.13	-0.5055	0.811	0.189	0.64	-0.2694	0.3389	0.6611
0.14	-0.4992	0.7985	0.2015	0.65	-0.2655	0.331	0.669
0.15	-0.493	0.7861	0.2139	0.66	-0.2615	0.3231	0.6769
0.16	-0.487	0.7739	0.2261	0.67	-0.2576	0.3151	0.6849
0.17	-0.481	0.762	0.238	0.68	-0.2536	0.3071	0.6929
0.18	-0.4751	0.7502	0.2498	0.69	-0.2495	0.2991	0.7009
0.19	-0.4693	0.7386	0.2614	0.7	-0.2455	0.291	0.709
0.2	-0.4636	0.7272	0.2728	0.71	-0.2414	0.2828	0.7172
0.21	-0.458	0.716	0.284	0.72	-0.2373	0.2746	0.7254
0.22	-0.4525	0.705	0.295	0.73	-0.2332	0.2664	0.7336
0.23	-0.4471	0.6941	0.3059	0.74	-0.229	0.258	0.742
0.24	-0.4417	0.6834	0.3166	0.75	-0.2248	0.2496	0.7504
0.25	-0.4365	0.6729	0.3271	0.76	-0.2206	0.2411	0.7589
0.26	-0.4313	0.6625	0.3375	0.77	-0.2163	0.2326	0.7674
0.27	-0.4262	0.6523	0.3477	0.78	-0.212	0.2239	0.7761
0.28	-0.4211	0.6422	0.3578	0.79	-0.2076	0.2152	0.7848
0.29	-0.4162	0.6323	0.3677	0.8	-0.2032	0.2063	0.7937
0.3	-0.4113	0.6225	0.3775	0.81	-0.1987	0.1974	0.8026
0.31	-0.4064	0.6129	0.3871	0.82	-0.1942	0.1883	0.8117
0.32	-0.4017	0.6034	0.3966	0.83	-0.1896	0.1791	0.8209
0.33	-0.397	0.594	0.406	0.84	-0.1849	0.1699	0.8301
0.34	-0.3923	0.5847	0.4153	0.85	-0.1802	0.1605	0.8395
0.35	-0.3878	0.5755	0.4245	0.86	-0.1755	0.1509	0.8491
0.36	-0.3832	0.5665	0.4335	0.87	-0.1706	0.1413	0.8587
0.37	-0.3788	0.5575	0.4425	0.88	-0.1657	0.1315	0.8685
0.38	-0.3743	0.5487	0.4513	0.89	-0.1608	0.1216	0.8784
0.39	-0.37	0.5399	0.4601	0.9	-0.1557	0.1115	0.8885
0.4	-0.3656	0.5313	0.4687	0.91	-0.1506	0.1013	0.8987
0.41	-0.3614	0.5227	0.4773	0.92	-0.1454	0.0909	0.9091
0.42	-0.3571	0.5142	0.4858	0.93	-0.1402	0.0804	0.9196
0.43	-0.3529	0.5058	0.4942	0.94	-0.1348	0.0697	0.9303
0.44	-0.3487	0.4975	0.5025	0.95	-0.1294	0.0588	0.9412
0.45	-0.3446	0.4892	0.5108	0.96	-0.1239	0.0478	0.9522
0.46	-0.3405	0.481	0.519	0.97	-0.1183	0.0365	0.9635
0.47	-0.3364	0.4729	0.5271	0.98	-0.1126	0.0252	0.9748
0.48	-0.3324	0.4648	0.5352	0.99	-0.1068	0.0136	0.9864
0.49	-0.3284	0.4567	0.5433	1	-0.1009	0.0018	0.9982
0.5	-0.3244	0.4488	0.5513				

Table B.4: Fuzzy Membership of *m* and *Sk* ($0 \leq SK \leq 1.0$)

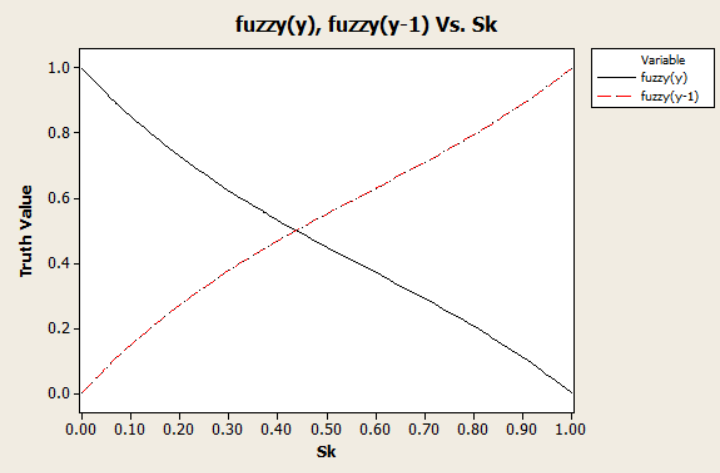

Fig.B.4: Fuzzy membership function for m=-0.1 to -0.6

B.5.2. Fuzzy Membership of *m* and *Sk* (-0.01 ≤ SK ≤ -0.52)

The objective of collecting real world data included in Table B.5 was filling the top row of the look-up table shown in Fig. B.5. Same procedure described in B.5.1 was adapted to fetch the Table B.5. Fig.B.6. is the final outcome.

Value required for estimation

y	0	?	?			*y*			?	?	1.00
Sk	-0.01	-0.02	-0.03			*Sk*			-0.50	-0.51	-0.52

Sk (Known / approximated)

Fig. B.5: The look-up table that keeps *y* values for minus *Sk*s

Sk	*m*	fuzzy(*y*)	fuzzy(*y*-1)
-0.01	-0.6039	0.0097	0.9903
-0.02	-0.6103	0.0259	0.9741
-0.03	-0.6169	0.0423	0.9577
-0.04	-0.6236	0.0589	0.9411
-0.05	-0.6303	0.0758	0.9242
-0.06	-0.6371	0.0929	0.9071
-0.07	-0.6441	0.1101	0.8899
-0.08	-0.651	0.1276	0.8724
-0.09	-0.6581	0.1453	0.8547
-0.1	-0.6653	0.1632	0.8369
-0.11	-0.6725	0.1812	0.8188
-0.12	-0.6798	0.1994	0.8006
-0.13	-0.6871	0.2178	0.7822
-0.14	-0.6945	0.2363	0.7637
-0.15	-0.702	0.255	0.745
-0.16	-0.7095	0.2738	0.7262
-0.17	-0.7171	0.2927	0.7073
-0.18	-0.7247	0.3118	0.6882
-0.19	-0.7324	0.331	0.669
-0.2	-0.7401	0.3504	0.6497
-0.21	-0.7479	0.3698	0.6302
-0.22	-0.7557	0.3893	0.6107
-0.23	-0.7636	0.4089	0.5911
-0.24	-0.7714	0.4286	0.5714

Sk	*m*	fuzzy(*y*)	fuzzy(*y*-1)
-0.27	-0.7953	0.4881	0.5119
-0.28	-0.8032	0.5081	0.4919
-0.29	-0.8112	0.5281	0.4719
-0.3	-0.8193	0.5482	0.4519
-0.31	-0.8273	0.5682	0.4318
-0.32	-0.8353	0.5883	0.4117
-0.33	-0.8434	0.6085	0.3915
-0.34	-0.8515	0.6286	0.3714
-0.35	-0.8595	0.6488	0.3512
-0.36	-0.8676	0.6689	0.3311
-0.37	-0.8756	0.6891	0.3109
-0.38	-0.8837	0.7092	0.2908
-0.39	-0.8917	0.7293	0.2707
-0.4	-0.8997	0.7494	0.2507
-0.41	-0.9078	0.7694	0.2306
-0.42	-0.9157	0.7894	0.2106
-0.43	-0.9237	0.8093	0.1907
-0.44	-0.9317	0.8292	0.1708
-0.45	-0.9396	0.849	0.151
-0.46	-0.9475	0.8687	0.1313
-0.47	-0.9553	0.8884	0.1116
-0.48	-0.9632	0.9079	0.0921
-0.49	-0.971	0.9274	0.0726
-0.5	-0.9787	0.9468	0.0533

-0.25	-0.7794	0.4484	0.5516	-0.51	-0.9864	0.966	0.034
-0.26	-0.7873	0.4682	0.5318	-0.52	-0.9941	0.9851	0.0149

Table B.5: Fuzzy membership function for m=-0.6 to -1.0

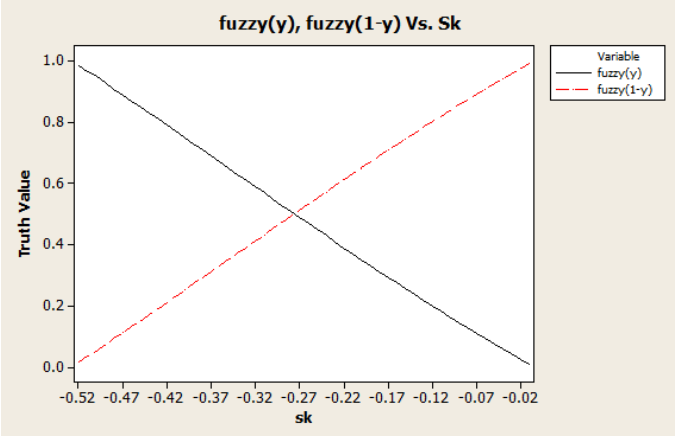

Fig.B.6: Fuzzy membership function for m=-0.6 to -1.0

B.5.3. Fuzzy Membership of *m* and *Sk* (-0.53 ≤ SK ≤ -3.0)

The objective of collecting real world data included in Table B.6 was filling the top row of the look-up table shown in Fig. B.7. Same procedure described in B.5.1 was adapted to fetch the Table B.6. Fig.B.8. is the final outcome.

y	0	?	?	*y*	?	?	1.00
Sk	-0.53	-0.54	-0.55	*Sk*	-2.98	-2.99	-3.00

Fig. B.7: The look-up table that keeps *y* values for larger minus *Sk*s

Sk	m	fuzzy(y)	fuzzy(1-y)
-0.53	-1.0312	0.0312	0.9688
-0.54	-1.038	0.038	0.962
-0.55	-1.0449	0.0449	0.9551
-0.56	-1.0517	0.0517	0.9483
-0.57	-1.0585	0.0585	0.9415
-0.58	-1.0653	0.0653	0.9347
-0.59	-1.072	0.072	0.928
-0.6	-1.0787	0.0787	0.9213
-0.61	-1.0854	0.0854	0.9146
-0.62	-1.0921	0.0921	0.9079
-0.63	-1.0987	0.0987	0.9013
-0.64	-1.1053	0.1053	0.8947
-0.65	-1.1119	0.1119	0.8881
-0.66	-1.1185	0.1185	0.8815
-0.67	-1.125	0.125	0.875
-0.68	-1.1315	0.1315	0.8685
-0.69	-1.138	0.138	0.862
-0.7	-1.1445	0.1445	0.8555
-0.71	-1.1509	0.1509	0.8491
-0.72	-1.1573	0.1573	0.8427
-0.73	-1.1637	0.1637	0.8363
-0.74	-1.1701	0.1701	0.8299
-0.75	-1.1764	0.1764	0.8236

Sk	m	fuzzy(y)	fuzzy(1-y)
-1.77	-1.689	0.689	0.311
-1.78	-1.6927	0.6927	0.3073
-1.79	-1.6964	0.6964	0.3036
-1.8	-1.7	0.7	0.3
-1.81	-1.7037	0.7037	0.2963
-1.82	-1.7073	0.7073	0.2927
-1.83	-1.7109	0.7109	0.2891
-1.84	-1.7144	0.7144	0.2856
-1.85	-1.718	0.718	0.282
-1.86	-1.7215	0.7215	0.2785
-1.87	-1.725	0.725	0.275
-1.88	-1.7284	0.7284	0.2716
-1.89	-1.7319	0.7319	0.2681
-1.9	-1.7353	0.7353	0.2647
-1.91	-1.7387	0.7387	0.2613
-1.92	-1.742	0.742	0.258
-1.93	-1.7454	0.7454	0.2546
-1.94	-1.7487	0.7487	0.2513
-1.95	-1.752	0.752	0.248
-1.96	-1.7552	0.7552	0.2448
-1.97	-1.7585	0.7585	0.2415
-1.98	-1.7617	0.7617	0.2383
-1.99	-1.7648	0.7648	0.2352

-0.76	-1.1827	0.1827	0.8173
-0.77	-1.189	0.189	0.811
-0.78	-1.1953	0.1953	0.8047
-0.79	-1.2015	0.2015	0.7985
-0.8	-1.2077	0.2077	0.7923
-0.81	-1.2139	0.2139	0.7861
-0.82	-1.22	0.22	0.78
-0.83	-1.2262	0.2262	0.7738
-0.84	-1.2323	0.2323	0.7677
-0.85	-1.2384	0.2384	0.7616
-0.86	-1.2444	0.2444	0.7556
-0.87	-1.2504	0.2504	0.7496
-0.88	-1.2564	0.2564	0.7436
-0.89	-1.2624	0.2624	0.7376
-0.9	-1.2684	0.2684	0.7316
-0.91	-1.2743	0.2743	0.7257
-0.92	-1.2802	0.2802	0.7198
-0.93	-1.2861	0.2861	0.7139
-0.94	-1.2919	0.2919	0.7081
-0.95	-1.2978	0.2978	0.7022
-0.96	-1.3036	0.3036	0.6964
-0.97	-1.3093	0.3093	0.6907
-0.98	-1.3151	0.3151	0.6849
-0.99	-1.3208	0.3208	0.6792
-1	-1.3265	0.3265	0.6735
-1.01	-1.3322	0.3322	0.6678
-1.02	-1.3378	0.3378	0.6622
-1.03	-1.3434	0.3434	0.6566
-1.04	-1.349	0.349	0.651
-1.05	-1.3546	0.3546	0.6454
-1.06	-1.3602	0.3602	0.6398
-1.07	-1.3657	0.3657	0.6343
-1.08	-1.3712	0.3712	0.6288
-1.09	-1.3766	0.3766	0.6234
-1.1	-1.3821	0.3821	0.6179
-1.11	-1.3875	0.3875	0.6125
-1.12	-1.3929	0.3929	0.6071
-1.13	-1.3983	0.3983	0.6017
-1.14	-1.4036	0.4036	0.5964
-1.15	-1.4089	0.4089	0.5911
-1.16	-1.4142	0.4142	0.5858
-1.17	-1.4195	0.4195	0.5805
-1.18	-1.4247	0.4247	0.5753
-1.19	-1.4299	0.4299	0.5701
-1.2	-1.4351	0.4351	0.5649
-1.21	-1.4403	0.4403	0.5597
-1.22	-1.4454	0.4454	0.5546
-1.23	-1.4506	0.4506	0.5494
-1.24	-1.4556	0.4556	0.5444
-1.25	-1.4607	0.4607	0.5393
-1.26	-1.4657	0.4657	0.5343
-1.27	-1.4708	0.4708	0.5292
-1.28	-1.4757	0.4757	0.5243
-1.29	-1.4807	0.4807	0.5193
-1.3	-1.4856	0.4856	0.5144
-1.31	-1.4906	0.4906	0.5094
-1.32	-1.4954	0.4954	0.5046
-1.33	-1.5003	0.5003	0.4997

-2	-1.768	0.768	0.232
-2.01	-1.7711	0.7711	0.2289
-2.02	-1.7742	0.7742	0.2258
-2.03	-1.7773	0.7773	0.2227
-2.04	-1.7804	0.7804	0.2196
-2.05	-1.7834	0.7834	0.2166
-2.06	-1.7864	0.7864	0.2136
-2.07	-1.7894	0.7894	0.2106
-2.08	-1.7923	0.7923	0.2077
-2.09	-1.7953	0.7953	0.2047
-2.1	-1.7982	0.7982	0.2018
-2.11	-1.801	0.801	0.199
-2.12	-1.8039	0.8039	0.1961
-2.13	-1.8067	0.8067	0.1933
-2.14	-1.8095	0.8095	0.1905
-2.15	-1.8123	0.8123	0.1877
-2.16	-1.8151	0.8151	0.1849
-2.17	-1.8178	0.8178	0.1822
-2.18	-1.8205	0.8205	0.1795
-2.19	-1.8231	0.8231	0.1769
-2.2	-1.8258	0.8258	0.1742
-2.21	-1.8284	0.8284	0.1716
-2.22	-1.831	0.831	0.169
-2.23	-1.8336	0.8336	0.1664
-2.24	-1.8361	0.8361	0.1639
-2.25	-1.8387	0.8387	0.1613
-2.26	-1.8412	0.8412	0.1588
-2.27	-1.8436	0.8436	0.1564
-2.28	-1.8461	0.8461	0.1539
-2.29	-1.8485	0.8485	0.1515
-2.3	-1.8509	0.8509	0.1491
-2.31	-1.8532	0.8532	0.1468
-2.32	-1.8556	0.8556	0.1444
-2.33	-1.8579	0.8579	0.1421
-2.34	-1.8602	0.8602	0.1398
-2.35	-1.8625	0.8625	0.1375
-2.36	-1.8647	0.8647	0.1353
-2.37	-1.8669	0.8669	0.1331
-2.38	-1.8691	0.8691	0.1309
-2.39	-1.8713	0.8713	0.1287
-2.4	-1.8734	0.8734	0.1266
-2.41	-1.8755	0.8755	0.1245
-2.42	-1.8776	0.8776	0.1224
-2.43	-1.8797	0.8797	0.1203
-2.44	-1.8817	0.8817	0.1183
-2.45	-1.8837	0.8837	0.1163
-2.46	-1.8857	0.8857	0.1143
-2.47	-1.8877	0.8877	0.1123
-2.48	-1.8896	0.8896	0.1104
-2.49	-1.8915	0.8915	0.1085
-2.5	-1.8934	0.8934	0.1066
-2.51	-1.8953	0.8953	0.1047
-2.52	-1.8971	0.8971	0.1029
-2.53	-1.8989	0.8989	0.1011
-2.54	-1.9007	0.9007	0.0993
-2.55	-1.9025	0.9025	0.0975
-2.56	-1.9042	0.9042	0.0958
-2.57	-1.9059	0.9059	0.0941

-1.34	-1.5051	0.5051	0.4949	-2.58	-1.9076	0.9076	0.0924
-1.35	-1.5099	0.5099	0.4901	-2.59	-1.9093	0.9093	0.0907
-1.36	-1.5147	0.5147	0.4853	-2.6	-1.9109	0.9109	0.0891
-1.37	-1.5195	0.5195	0.4805	-2.61	-1.9125	0.9125	0.0875
-1.38	-1.5242	0.5242	0.4758	-2.62	-1.9141	0.9141	0.0859
-1.39	-1.5289	0.5289	0.4711	-2.63	-1.9156	0.9156	0.0844
-1.4	-1.5336	0.5336	0.4664	-2.64	-1.9172	0.9172	0.0828
-1.41	-1.5383	0.5383	0.4617	-2.65	-1.9187	0.9187	0.0813
-1.42	-1.5429	0.5429	0.4571	-2.66	-1.9201	0.9201	0.0799
-1.43	-1.5475	0.5475	0.4525	-2.67	-1.9216	0.9216	0.0784
-1.44	-1.5521	0.5521	0.4479	-2.68	-1.923	0.923	0.077
-1.45	-1.5566	0.5566	0.4434	-2.69	-1.9244	0.9244	0.0756
-1.46	-1.5612	0.5612	0.4388	-2.7	-1.9258	0.9258	0.0742
-1.47	-1.5657	0.5657	0.4343	-2.71	-1.9272	0.9272	0.0728
-1.48	-1.5701	0.5701	0.4299	-2.72	-1.9285	0.9285	0.0715
-1.49	-1.5746	0.5746	0.4254	-2.73	-1.9298	0.9298	0.0702
-1.5	-1.579	0.579	0.421	-2.74	-1.9311	0.9311	0.0689
-1.51	-1.5834	0.5834	0.4166	-2.75	-1.9323	0.9323	0.0677
-1.52	-1.5878	0.5878	0.4122	-2.76	-1.9335	0.9335	0.0665
-1.53	-1.5922	0.5922	0.4078	-2.77	-1.9347	0.9347	0.0653
-1.54	-1.5965	0.5965	0.4035	-2.78	-1.9359	0.9359	0.0641
-1.55	-1.6008	0.6008	0.3992	-2.79	-1.9371	0.9371	0.0629
-1.56	-1.6051	0.6051	0.3949	-2.8	-1.9382	0.9382	0.0618
-1.57	-1.6093	0.6093	0.3907	-2.81	-1.9393	0.9393	0.0607
-1.58	-1.6135	0.6135	0.3865	-2.82	-1.9403	0.9403	0.0597
-1.59	-1.6177	0.6177	0.3823	-2.83	-1.9414	0.9414	0.0586
-1.6	-1.6219	0.6219	0.3781	-2.84	-1.9424	0.9424	0.0576
-1.61	-1.6261	0.6261	0.3739	-2.85	-1.9434	0.9434	0.0566
-1.62	-1.6302	0.6302	0.3698	-2.86	-1.9444	0.9444	0.0556
-1.63	-1.6343	0.6343	0.3657	-2.87	-1.9453	0.9453	0.0547
-1.64	-1.6383	0.6383	0.3617	-2.88	-1.9462	0.9462	0.0538
-1.65	-1.6424	0.6424	0.3576	-2.89	-1.9471	0.9471	0.0529
-1.66	-1.6464	0.6464	0.3536	-2.9	-1.948	0.948	0.052
-1.67	-1.6504	0.6504	0.3496	-2.91	-1.9489	0.9489	0.0511
-1.68	-1.6544	0.6544	0.3456	-2.92	-1.9497	0.9497	0.0503
-1.69	-1.6583	0.6583	0.3417	-2.93	-1.9505	0.9505	0.0495
-1.7	-1.6622	0.6622	0.3378	-2.94	-1.9512	0.9512	0.0488
-1.71	-1.6661	0.6661	0.3339	-2.95	-1.952	0.952	0.048
-1.72	-1.67	0.67	0.33	-2.96	-1.9527	0.9527	0.0473
-1.73	-1.6738	0.6738	0.3262	-2.97	-1.9534	0.9534	0.0466
-1.74	-1.6777	0.6777	0.3223	-2.98	-1.954	0.954	0.046
-1.75	-1.6815	0.6815	0.3185	-2.99	-1.9547	0.9547	0.0453
-1.76	-1.6852	0.6852	0.3148	-3	-1.9553	0.9553	0.0447

Table B.6: Fuzzy membership function for m=-1.0 to -2.0

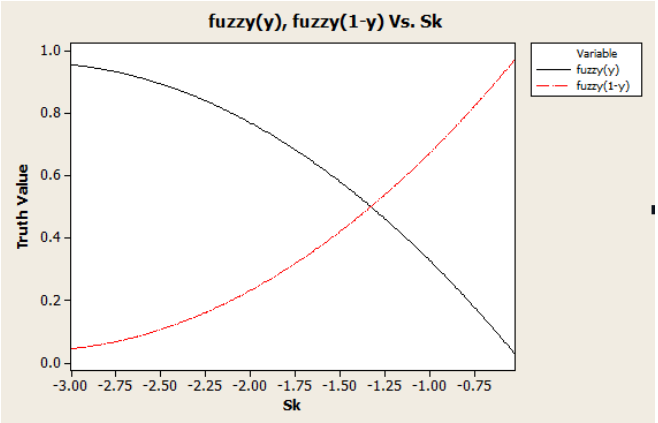

Fig.B.8: Fuzzy membership function for gradients lesser than -1.0

B.5.4. Fuzzy Membership of Intersection(c), Gradient(m) and Initial Apparent Size

This relationship is bit complicated compared to gradient-skewness relationship. The strategy followed to realize that was as follows:

The variation of the intersect (c) and the initial apparent size of the object was studied while maintaining the gradient (m) constant at a particular instance. Then again m was set to a different value, kept constant and repeated the experiment. The following formulae were used to find *y*.

$$c = (y * H) + [(1 - y) * L]$$

$$c(m = -0.1) = (y * 606) + [(1 - y) * 89]$$

$$\Rightarrow y = (c - 89) / 517$$

$$c(m = -1.0) = (y * 317) + [(1 - y) * 39]$$

$$\Rightarrow y = (c - 39) / 278$$

Table B.7. shows the experimental results. Fig.B.9 illustrates the final outcome.

initial size	*c*, *m*=-0.1	*y*	1-*y*
10	606	1	0
11	577	0.9439	0.0561
12	546	0.8839	0.1161
13	516	0.8259	0.1741
14	485	0.766	0.234
15	456	0.7099	0.2901
16	442	0.6828	0.3172
17	427	0.6538	0.3462
18	397	0.5957	0.4043
19	379	0.5609	0.4391
20	365	0.5338	0.4662
21	348	0.501	0.499
22	335	0.4758	0.5242
23	328	0.4623	0.5377
24	317	0.441	0.559
25	304	0.4159	0.5841
26	292	0.3926	0.6074
27	284	0.3772	0.6228
28	274	0.3578	0.6422
29	269	0.3482	0.6518
30	259	0.3288	0.6712

initial size	*c*, *m*=-1.0	*y*	1-*y*
10	317	1	0
11	295	0.9209	0.0791
12	285	0.8849	0.1151
13	267	0.8201	0.1799
14	257	0.7842	0.2158
15	246	0.7446	0.2554
16	228	0.6799	0.3201
17	219	0.6475	0.3525
18	204	0.5935	0.4065
19	194	0.5576	0.4424
20	193	0.554	0.446
21	186	0.5288	0.4712
22	170	0.4712	0.5288
23	166	0.4568	0.5432
24	159	0.4317	0.5683
25	154	0.4137	0.5863
26	148	0.3921	0.6079
27	145	0.3813	0.6187
28	138	0.3561	0.6439
29	136	0.3489	0.6511
30	133	0.3381	0.6619

31	255	0.3211	0.6789	31	126	0.3129	0.6871
32	248	0.3075	0.6925	32	123	0.3022	0.6978
33	243	0.2979	0.7021	33	118	0.2842	0.7158
34	233	0.2785	0.7215	34	115	0.2734	0.7266
35	231	0.2747	0.7253	35	115	0.2734	0.7266
36	220	0.2534	0.7466	36	111	0.259	0.741
37	218	0.2495	0.7505	37	110	0.2554	0.7446
38	210	0.234	0.766	38	104	0.2338	0.7662
39	206	0.2263	0.7737	39	103	0.2302	0.7698
40	201	0.2166	0.7834	40	101	0.223	0.777
41	198	0.2108	0.7892	41	96	0.205	0.795
42	193	0.2012	0.7988	42	95	0.2014	0.7986
43	185	0.1857	0.8143	43	93	0.1942	0.8058
44	184	0.1838	0.8162	44	90	0.1835	0.8165
45	179	0.1741	0.8259	45	88	0.1763	0.8237
46	177	0.1702	0.8298	46	85	0.1655	0.8345
47	174	0.1644	0.8356	47	83	0.1583	0.8417
48	169	0.1547	0.8453	48	82	0.1547	0.8453
49	165	0.147	0.853	49	80	0.1475	0.8525
50	163	0.1431	0.8569	50	78	0.1403	0.8597
51	160	0.1373	0.8627	51	77	0.1367	0.8633
52	157	0.1315	0.8685	52	76	0.1331	0.8669
53	155	0.1277	0.8723	53	75	0.1295	0.8705
54	151	0.1199	0.8801	54	72	0.1187	0.8813
55	148	0.1141	0.8859	55	71	0.1151	0.8849
56	146	0.1103	0.8897	56	71	0.1151	0.8849
57	144	0.1064	0.8936	57	70	0.1115	0.8885
58	141	0.1006	0.8994	58	68	0.1043	0.8957
59	139	0.0967	0.9033	59	67	0.1007	0.8993
60	136	0.0909	0.9091	60	66	0.0971	0.9029
61	133	0.0851	0.9149	61	64	0.0899	0.9101
62	131	0.0812	0.9188	62	63	0.0863	0.9137
63	129	0.0774	0.9226	63	61	0.0791	0.9209
64	127	0.0735	0.9265	64	60	0.0755	0.9245
65	124	0.0677	0.9323	65	59	0.0719	0.9281
66	123	0.0658	0.9342	66	57	0.0647	0.9353
67	122	0.0638	0.9362	67	54	0.054	0.946
68	121	0.0619	0.9381	68	54	0.054	0.946
69	118	0.0561	0.9439	69	55	0.0576	0.9424
70	117	0.0542	0.9458	70	53	0.0504	0.9496
71	116	0.0522	0.9478	71	52	0.0468	0.9532
72	114	0.0484	0.9516	72	51	0.0432	0.9568
73	112	0.0445	0.9555	73	50	0.0396	0.9604
74	111	0.0426	0.9574	74	50	0.0396	0.9604
75	109	0.0387	0.9613	75	50	0.0396	0.9604
76	107	0.0348	0.9652	76	49	0.036	0.964
77	105	0.0309	0.9691	77	49	0.036	0.964
78	103	0.0271	0.9729	78	48	0.0324	0.9676
79	102	0.0251	0.9749	79	47	0.0288	0.9712
80	101	0.0232	0.9768	80	47	0.0288	0.9712
81	100	0.0213	0.9787	81	44	0.018	0.982
82	99	0.0193	0.9807	82	43	0.0144	0.9856
83	97	0.0155	0.9845	83	43	0.0144	0.9856
84	96	0.0135	0.9865	84	42	0.0108	0.9892
85	95	0.0116	0.9884	85	42	0.0108	0.9892
86	93	0.0077	0.9923	86	43	0.0144	0.9856
87	92	0.0058	0.9942	87	42	0.0108	0.9892
88	91	0.0039	0.9961	88	41	0.0072	0.9928

89	89	0	1	89	40	0.0036	0.9964
90	89	0	1	90	39	0	1

Table B.7. Fuzzy membership function of *c* with initial size, when *m*=-0.1 and m=-1

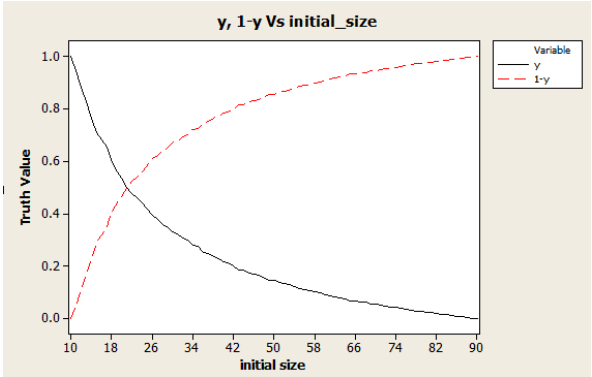

Fig.B.9: Fuzzy membership function of *c* with initial size

Appendix C:

Code Level Implementation of FMM

C.1. Introduction

This appendix includes the source code of the key implementations of the Fuzzy Mathematical Model (Real-Time version) with C#.Net programming language.

C.2. Implementation of FMM-Simulator Version

Below (Code Fragment C.1) is the source code of the Fuzzy Mathematical Model, implemented with C#.Net (.Net Version 4.5 with IDE-Visual Studio 2012 Professional).

Please note that some non-innovative code fragments (i.e. graphical illustrations such as graph plotting) were intentionally left due to the reason avoiding the appendix get unnecessarily lengthy. Please see the soft-copy available on the Compact Disk provided with for full source code.


```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Drawing.Drawing2D;

namespace FuzzyModelingSim
{
    public partial class Form1 : Form
    {
        int[] dist = new int[300];
        int[] posDiv = new int[300];
        int[] linear = new int[300];
        double[] apprSize = new double[300];
        double skewness = 0;
        int initSize = 0;
        double gradient = 0.0;
        int intersect = 0;
        double[] sk0To1=new double[101];
        double[] sk0M52To0 = new double[52];
        double[] sk3MTo0M52 = new double[248];
        double[] c0M1 = new double[81];
        double[] cM1 = new double[82];


        void plotGraphSize()
        {
              //Recover (and draw) initial size variation graph
        }
```

```
//Calculate Skewness of apparent size variation curve
 void calcSkewness()
 {
     //mean
     double mean = 0.0;
     for (int i = 0; i < 300; i++)
     {
         mean += apprSize[i];
     }
     mean /= 300;

     double diviationSq = 0.0;
     double diviationCube = 0.0;
     for (int i = 0; i < 300; i++)
     {
         diviationSq += ((mean - apprSize[i]) * (mean - apprSize[i]));
         diviationCube += ((mean - apprSize[i]) * (mean - apprSize[i]) * (mean
             - apprSize[i]));
     }
     double s = Math.Sqrt((1.0 / 299.0) * diviationSq);

     skewness = diviationCube / (299.0 * s * s * s);
     lblSkewness.Text = skewness.ToString("N4");
 }

 //Recover (and draw) Position deviation graph
 void plotGraphPosDiv()
 {

 }

 //Read Clipboard and load Graphs
 private void btnGet_Click(object sender, EventArgs e)
 {
     try
     {
       ..........................................
       ..........................................
             plotGraphSize();
             calcSkewness();
             plotGraphPosDiv();
     }
     catch (Exception ex)
     {
         ......................................
     }
 }

 //Fill Fuzzy Lookup tables from saved files
 void loadFuzzyLookUps()
 {
    //Read Data file for Sk=0 ~ 1.00
     try
     {
         string[] lines = System.IO.File.ReadAllLines("SK_0_to_1.txt");
         int i = 0;
         foreach (string line in lines)
         {
             if(i<=100)
                 sk0To1[i] = double.Parse(line);
             i++;
         }
```

```
            string[] lines2 = System.IO.File.ReadAllLines("SK_M1_to_0.txt");
            i = 0;
            foreach (string line in lines2)
            {
                if (i <= 52)
                    sk0M52To0[i] = double.Parse(line);
                i++;
            }

            string[] lines3 = System.IO.File.ReadAllLines("SK_M2_to_M1.txt");
            i = 0;
            foreach (string line in lines3)
            {
                if (i <= 248)
                    sk3MTo0M52[i] = double.Parse(line);
                i++;
            }

            string[] lines4 = System.IO.File.ReadAllLines("C_Size_0M1.txt");
            i = 0;
            foreach (string line in lines4)
            {
                if (i <= 80)
                    c0M1[i] = double.Parse(line);
                i++;
            }
            string[] lines5 = System.IO.File.ReadAllLines("C_Size_M1.txt");
            i = 0;
            foreach (string line in lines5)
            {
                if (i <= 81)
                cM1[i] = double.Parse(line);
                i++;
            }

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }

    }


    //Fuzzy Modeling (Implementation of FMM)
    private void btnAnalyze_Click(object sender, EventArgs e)
    {
        loadFuzzyLookUps();

        //Size @ Entry Position :
        initSize = (int)apprSize[1];
        lblSize.Text = "Size @ Entry Position : " + initSize.ToString();

        //Fuzzy based estimations
        double grad=0.0;
        //For Sk 0~1 (-0.6<= m <=-0.1)
        if(skewness >= 0 && skewness <= 1)
        {
        //find matching index of array by sk value
        //0-->0, 0.01-->1, 0.02-->2...0.98-->98...ect.
            int index = Convert.ToInt32(skewness*100);
```

```
            double y = sk0To1[index];
            //m = [y*(-0.6)] + [(1-y)*(-0.1)]
            grad = (y * (-0.6)) + ((1 - y) * (-0.1));
            gradient = grad;
        }
        //For Sk -0.52~0 (-1.0<= m <=-0.6)
        if (skewness >= -0.52 && skewness < 0)
        {
            //find matching index of array by sk value
            //-0.01-->0, -0.02-->1, -0.03-->2...-0.52-->51...ect.
            int index = Convert.ToInt32(-1.0*skewness * 100.0)-1;
            double y = sk0M52To0[index];
            //m = [y*(-1.0)] + [(1-y)*(-0.6)]
            grad = (y * (-1.0)) + ((1 - y) * (-0.6));
            gradient = grad;
        }
        //For Sk -3.0~-0.53 (-2.0<= m <=-1.0)
        if (skewness >= -3.0 && skewness < -0.52)
        {
            //find matching index of array by sk value
            //-0.53-->0, -0.54-->1, -0.55-->2...-3.00-->247...ect.
            int index = Convert.ToInt32(-1.0 * skewness * 100.0) - 53;
            double y = sk3MTo0M52[index];
            //m = [y*(-2.0)] + [(1-y)*(-1.0)]
            grad = (y * (-2.0)) + ((1 - y) * (-1.0));
            gradient = grad;
        }

        //Fuzzy based estimation for Intersection (c)
        double finalC = 0.0;
        if (grad <= -0.1 && grad >= -1.0)
        {
            //find matching index of array by c value
            //initSize=10-->0, 11-->1, 12-->2...-90-->80...ect.
            int index = initSize - 10;
            double y=0.0;
            if(index<=80)
                y = c0M1[index];
            double y3 = 0.0;
            if (index <= 81)
                y3 = cM1[index];

            lblFuzzyGrad.Text = "c (initSize) = " + y.ToString("N2") + " H + " +
                (1 - y).ToString("N2") + " L";

            //c = ( y * H )+[ (1 - y) * L ]
            int c1 = Convert.ToInt32((y * 606.0) + ((1 - y) * 89.0));
            lblFuzzyC0.Text = "c1 = " + c1.ToString();
            int c2 = Convert.ToInt32((y3 * 317.0) + ((1 - y3) * 39.0));
            lblC1.Text = "c2 = " + c2.ToString();

            //c1:c2 ratio from m
            double y2 = (1.0 - Math.Abs(grad)) / 0.99;
            lblC2.Text = "m = " + (1 - y2).ToString("N2") + " H + " +
                y2.ToString("N2") + " L";

            finalC = c2 + (Math.Abs(c1 - c2) * (y2));
            intersect = (int)finalC;
        }
```

```
            if (grad <= -1.0 && grad >= -2.0)
            {
                //find matching index of array by c value
                //initSize=10-->0, 11-->1, -12-->2...-90-->80...ect.
                int index = initSize - 10;

                double y = 0.0;
                if (index <= 81)
                    y = cM1[index];

                lblFuzzyGrad.Text = "c (initSize) = " + y.ToString("N2") + " H + " +
                     (1 - y).ToString("N2") + " L";

                //c = ( y * H )+[ (1 - y) * L ]
                int c1 = Convert.ToInt32((y * 317.0) + ((1 - y) * 39.0));
                lblFuzzyC0.Text = "c1 = " + c1.ToString();
                lblC1.Text = "c2 = 0";

                //c1:c2 ratio from m
                double y2 = (2.0 - Math.Abs(grad));
                lblC2.Text = "m = " + (1 - y2).ToString("N2") + " H + " +
                     y2.ToString("N2") + " L";

                //finalC = c2 + (Math.Abs(c1 - c2) * (y2));
                finalC = (Math.Abs(c1) * (y2));
                intersect = (int)finalC;
            }

            lblPath.Text = "y = " + grad.ToString("N2") + " x + " +
              finalC.ToString("N0");

            drawEstimated();
            drawPathRobot();
        }

        //Draw Predicted Path of the object on coordinate system
        void drawEstimated()
        {

        }
    }
}
```


Code Fragment C.1: Source code of the Fuzzy Mathematical Model

C.3. Implementation of FMM: Real-Time Version

Below (Code Fragment C.2) is the source code of the Fuzzy Mathematical Model (Real-Time Version), implemented with C#.Net (.Net Version 4.5 with IDE-Visual Studio 2012 Professional).

Please note that some non-innovative code fragments were intentionally left due to the reason avoiding the appendix get unnecessarily lengthy. Please see the soft-copy available on the Compact Disk provided with for full source code.

```
using //Same namespaces used in Simulator version;

namespace FuzzyModelingRT
{
    public partial class Form1 : Form
    {
        int currSize = 0;
        int prevSize = 0;
        int count = 0;

       //Same variables used in Simulator version

       // Create new stopwatch
        Stopwatch stopwatch = new Stopwatch();

        public Form1()
        {
            InitializeComponent();
        }

        void plotGraphSize()
        {

        }

        void plotGraphPosDiv()
        {

        }

        void calcSkewness()
        {
             //Implementation Same as Simulator version
        }

        void fuzzyModeling()
        {
            //Implementation Same as btnAnalyze_Click event in Simulator version
        }

        void drawEstimated()
        {

        }

        //Fill Fuzzy Lookup tables from saved files
        void loadFuzzyLookUps()
        {
           //Implementation Same as in Simulator version
        }


        private void Form1_Load(object sender, EventArgs e)
        {
            loadFuzzyLookUps();
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            try
            {
```

```
                currSize = int.Parse(Clipboard.GetText());//Continuously Reading
                     Clipboard to obtain output from Image Processing Module
                lblCurrSize.Text = currSize.ToString();

                if (currSize != prevSize)
                {
                    if (prevSize == 0)//Very 1st Reading
                    {
                        initSize = currSize;
                        // Begin timing
                        stopwatch.Start();
                        lblStopWatch.ForeColor = Color.Aqua;
                    }
                    if (count < 300)
                        size[count] = currSize;
                    count++;
                    TimeSpan ts = stopwatch.Elapsed;
                    // Format and display the TimeSpan value.

                    //Draw Size Graph
                    plotGraphSize();

                    //Fuzzy Modeling
                    fuzzyModeling();

                    prevSize = currSize; //Update
                }
            }
            catch (Exception ex)
            {
                .....
            }
        }
    }
}
```


Code Fragment C.2: Source code of the Fuzzy Mathematical Model -Real Time Edition

Appendix D:

Evaluation Results - Full Detailed View

D.1. Introduction

This appendix includes the evaluation results of each module developed under the implementation of software artifact to test the hypothesis, in details. Only a summary of these descriptive experimental data and graphical representation of key implications were presented in the evaluation chapter of the dissertation.

D.2. Evaluation of FMM-Simulator Version

Since most of the experiments done for testing improving and fine-tuning the FMM using synthetic data generated by the simulator, it is worth to verify the accuracy of it in order to evaluate the accuracy of other dependent modules. Table D.1. presents the relevant evaluation results.

initial size	Actual m	Actual c	Estimated m	Estimated c	m Error	c Error	m Error %	c Error %
10	-0.11	604	-0.1	574	-0.01	30	9.09	4.97
	-0.19	576	-0.19	548	0	28	2.06	4.86
	-0.33	531	-0.34	504	0.01	27	-2.1	5.08
	-0.38	515	-0.38	491	0	24	0.52	4.66
	-0.53	469	-0.53	450	0	19	-0.57	4.05
	-0.61	442	-0.62	423	0.01	19	-1.47	4.3
	-0.82	378	-0.81	365	-0.01	13	0.61	3.44
	-0.97	334	-0.95	325	-0.02	9	1.55	2.69
	-1.14	275	-1.14	265	0	10	0.09	3.64
	-1.22	248	-1.21	244	-0.01	4	0.66	1.61
	-1.3	220	-1.29	220	-0.01	0	0.92	0
	-1.44	179	-1.41	183	-0.03	-4	1.74	-2.23
	-1.58	135	-1.56	137	-0.02	-2	1.27	-1.48
	-1.8	65	-1.83	53	0.03	12	-1.89	18.46
16	-0.11	474	-0.1	464	-0.01	10	7.41	2.11
	-0.2	408	-0.2	400	0	8	1.96	1.96
	-0.37	360	-0.37	363	0	-3	-0.54	-0.83
	-0.66	319	-0.57	322	-0.09	-3	13.11	-0.94
	-0.8	265	-0.79	264	-0.01	1	0.75	0.38
	-0.95	219	-0.96	228	0.01	-9	-1.05	-4.11
	-1.33	147	-1.32	156	-0.01	-9	0.98	-6.12
	-1.72	59	-1.73	59	0.01	0	-0.46	0
	-1.81	38	-1.88	27	0.07	11	-3.98	28.95
	-1.85	25	-1.94	13	0.09	12	-4.81	48
25	-0.15	294	-0.13	285	-0.02	9	13.33	3.06
	-0.24	282	-0.25	279	0.01	3	-5.49	1.06
	-0.29	273	-0.3	261	0.01	12	-4.53	4.4
	-0.54	232	-0.55	223	0.01	9	-2.61	3.88
	-0.67	215	-0.67	205	0	10	-0.45	4.65
	-0.82	187	-0.82	181	0	6	0.12	3.21
	-0.99	169	-0.98	169	-0.01	0	0.71	0
	-1.12	141	-1.11	137	-0.01	4	0.54	2.84
	-1.38	104	-1.32	108	-0.06	-4	4	-3.85

	-1.54	71	-1.51	76	-0.03	-5	2.01	-7.04
	-1.73	38	-1.77	35	0.04	3	-2.37	7.89
	-1.81	25	-1.87	20	0.06	5	-3.49	20
45	-0.14	179	-0.13	173	-0.01	6	9.09	3.35
	-0.27	164	-0.27	159	0	5	-1.12	3.05
	-0.33	159	-0.34	152	0.01	7	-2.1	4.4
	-0.5	144	-0.49	138	-0.01	6	2	4.17
	-0.62	131	-0.62	126	0	5	-0.81	3.82
	-0.7	123	-0.7	121	0	2	0	1.63
	-0.87	104	-0.88	101	0.01	3	-1.15	2.88
	-0.94	100	-0.93	99	-0.01	1	0.64	1
	-1.02	91	-1.04	87	0.02	4	-1.66	4.4
	-1.2	76	-1.17	75	-0.03	1	2.09	1.32
	-1.33	59	-1.32	60	-0.01	-1	0.38	-1.69
	-1.65	24	-1.7	27	0.05	-3	-3.09	-12.5
	-1.72	18	-1.75	22	0.03	-4	-1.63	-22.22
70	-0.11	116	-0.12	110	0.01	6	-5.26	5.17
	-1.19	110	-1.18	106	-0.01	4	0.84	3.64
	-0.32	98	-0.34	93	0.02	5	-7.59	5.1
	-0.51	84	-0.53	81	0.02	3	-3.11	3.57
	-0.59	71	-0.62	68	0.03	3	-4.38	4.23
	-0.77	66	-0.76	63	-0.01	3	0.91	4.55
	-1	48	-1.07	45	0.07	3	-7	6.25
	-1.31	31	-1.37	32	0.06	-1	-4.74	-3.23
	-1.4	22	-1.47	27	0.07	-5	-5	-22.73
	-1.55	13	-1.59	18	0.04	-5	-2.91	-38.46

Table D.2. Accuracy of the Simulator

D.3.1 Evaluation of FMM - Real Time Version by Accuracy Reported After a Definite Time

The evaluation results of FMM - real time version by accuracy reported after a definite time are included in Table D.3.

D.3.2 Evaluation of FMM - Real Time Version by Time Elapsed to Achieve an Accuracy Level

The evaluation results of FMM - real time version by accuracy reported to achieve an accuracy level are included in Table D.4.

Actual *m*	Actual *c* (cm)	initialSize	Reading after *x* seconds								Accuracy %			
			3s		5s		10s		15s		10s		15s	
			m	c	m	c	m	c	m	c	m	c	m	c
-0.1	425	16	-1.89	0	-0.93	18	-0.19	127	-0.17	313	10	29.88	30	73.65
-0.5	325	16	-1.86	0	-0.99	13	-0.71	93	-0.39	279	58	28.62	78	85.85
-1	220	16	-1.79	0	-1.66	9	-1.23	74	-0.91	161	77	33.64	91	73.18
-1.5	100	16	-1.78	0	-1.71	8	-1.68	31	-1.59	63	88	31	94	63
-2	0	16	-1.76	0	-1.74	0	-1.74	0	-1.73	0	87	0	86.5	0
-0.1	300	25	-1.83	0	-0.97	17	-0.21	99	-0.17	178	0	33	30	59.33
-0.5	230	25	-1.81	0	-0.91	11	-0.66	87	-0.31	112	68	37.83	62	48.7
-1	150	25	-1.76	0	-1.33	6	-1.23	44	-0.88	93	77	29.33	88	62
-1.5	75	25	-1.76	0	-1.74	0	-1.69	8	-1.23	37	87.33	10.67	82	49.33
-2	0	25	-1.7	0	-1.53	0	-1.49	0	Stopped	Stopped	74.5	0	0	0
-0.1	175	45	-1.76	0	-1.49	11	-0.23	91	-0.18	131	0	52	20	74.86
-0.5	140	45	-1.76	0	-1.62	7	-0.67	59	-0.44	97	66	42.14	88	69.29
-1	85	45	-1.76	0	-1.63	5	-0.92	19	-1.36	53	92	22.35	64	62.35
-1.5	40	45	-1.73	0	-1.61	0	-1.65	6	-1.58	21	90	15	94.67	52.5
-2	0	45	-1.59	0	-1.59	0	Stopped	Stopped	Stopped	Stopped	0	0	0	0
-0.1	120	70	-1.76	0	-1.76	2	-1.41	27	-0.19	59	0	22.5	10	49.17
-0.5	90	70	-1.76	0	-1.76	0	-1.59	14	-0.68	47	0	15.56	64	52.22
-1	55	70	-1.75	0	-1.67	0	-1.63	3	Stopped	Stopped	37	5.45	0	0
-1.5	20	70	-1.74	0	-1.54	0	Stopped	Stopped	Stopped	Stopped	0	0	0	0
-2	0	70	0	0	Stopped	Stopped	Stopped	Stopped	Stopped	Stopped	0	0	0	0
-0.1	90	90	-1.76	0	-1.76	1	-1.61	19	-0.2	56	0	21.11	0	62.22
-0.5	70	90	-1.76	0	-1.76	0	-1.77	7	-1	30	0	10	0	42.86
-1	40	90	0	0	-1.76	0	Stopped	Stopped	Stopped	Stopped	0	0	0	0
-1.5	15	90	0	0	Stopped	Stopped	Stopped	Stopped	Stopped	Stopped	0	0	0	0
-2	0	90	0	0	Stopped	Stopped	Stopped	Stopped	Stopped	Stopped	0	0	0	0

Table D.3: Accuracy of the FMM-Real Time Application - Accuracy reported after *x* seconds

* Stopped : Obstacle Interrupted / Intentionally stopped the robot to avoid collision(obstacle too close to robot)

Actual *m*	Actual *c*(cm)	initialSize	Time Elapsed to achieve the Accuracy Level - Both *m* & *c*				Time Elapsed to achieve Accuracy Level - c only			
			60%	70%	80%	90%	60%	70%	80%	90%
-0.1	425	16	18.84	Never	Never	Never	8.47	13.8	15.1	17.89
-0.5	325	16	10.47	13.12	15.64	18.25	7.11	11.42	13.38	16.65
-1	220	16	9.93	12.98	14.11	17.41	8.39	13.94	16.21	17.61
-1.5	100	16	11.08	13.81	14.67	19.02	10.88	13.61	14.28	18.73
-2	0	16	Never	Never	Never	Never	Never	Never	Never	Never
-0.1	300	25	19.92	Never	Never	Never	15.06	16.83	Never	Never
-0.5	230	25	15.74	18.23	Never	Never	14.28	16.12	Never	Never
-1	150	25	13.38	15.51	16.22	Never	14.11	15.78	16.14	Never
-1.5	75	25	Never	Never	Never	Never	15.99	Never	Never	Never
-2	0	25	Never	Never	Never	Never	Never	Never	Never	Never
-0.1	175	45	Never	Never	Never	Never	10.56	13.32	14.99	17.53
-0.5	140	45	12.89	15.08	17.72	18.23	12.28	15.02	16.46	17.95
-1	85	45	13.8	16.62	19.76	Never	13.88	16.01	19.02	Never
-1.5	40	45	15.56	17.95	Never	Never	14.69	16.33	Never	Never
-2	0	45	Never	Never	Never	Never	Never	Never	Never	Never
-0.1	120	70	Never	Never	Never	Never	16.54	Never	Never	Never
-0.5	90	70	16.02	18.31	Never	Never	15.8	18.04	Never	Never
-1	55	70	Never	Never	Never	Never	Never	Never	Never	Never
-1.5	20	70	Never	Never	Never	Never	Never	Never	Never	Never
-2	0	70	Never	Never	Never	Never	Never	Never	Never	Never
-0.1	90	90	Never	Never	Never	Never	14.12	15.98	Never	Never
-0.5	70	90	Never	Never	Never	Never	16.01	Never	Never	Never
-1	40	90	Never	Never	Never	Never	Never	Never	Never	Never
-1.5	15	90	Never	Never	Never	Never	Never	Never	Never	Never
-2	0	90	Never	Never	Never	Never	Never	Never	Never	Never

Table D.4: Accuracy of the FMM-Real Time Application- Time Elapsed to achieve an accuracy level

Appendix E:

Source Code of the Robot

E.1. Introduction

This appendix includes the instructions specified for the robot, written in C programming language. The code is as Code Fragment E.1.

```
// include the library code:
#include <LiquidCrystal.h>
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

const int buffersize = 64;

unsigned char buffer[buffersize]; //64 character buffer

void setup()
{
	Serial.begin(9600);

	// set up the LCD's number of columns and rows:
	lcd.begin(16, 2);
	// Print a message to the LCD.
	lcd.setCursor(0,0);
	lcd.print("I AM ARTIFICIAL");
	lcd.setCursor(0,1);
	lcd.print("INTELLIGENCE LAB");

	pinMode(52, OUTPUT);//Enable A
	pinMode(53, OUTPUT);//Enable A
	pinMode(50, OUTPUT);//Motor A In1
	pinMode(51, OUTPUT);//Motor A In1
	pinMode(48, OUTPUT);//Motor A In2
	pinMode(49, OUTPUT);//Motor A In2
	pinMode(42, OUTPUT);//Enable B
	pinMode(43, OUTPUT);//Enable B
	pinMode(46, OUTPUT);//Motor B In3
	pinMode(47, OUTPUT);//Motor B In3
	pinMode(44, OUTPUT);//Motor B In4
	pinMode(45, OUTPUT);//Motor B In4

	//Encoders
	pinMode(24, INPUT);//Encoder2
}

//Flags
int counter=0;
boolean isStarted=false;
boolean isStopped=false;
boolean isRotated=false;
boolean avoidObs=false;
boolean testRun=false;
boolean done=false;

int rorateAngle=0;
boolean isCounterClockWise=false;
```

```
float riskFactor=0.0;

void loop()
{
	if(Serial.available() > 0)

	{
		// wait a bit for the entire message to arrive
		delay(100);
		// clear the screen
		lcd.clear();

		int i = 0;
		unsigned char data = 0;

		while(Serial.available() > 0)
		{
			data = Serial.read();

			if(i < buffersize)
			{
				buffer[i] = data; //put all data into buffer until if
fills

				i++;
			}

		}

		//Obtaining rorateAngle from buffer
		char rAngle[2];
		rAngle[0] = buffer[0];
		rAngle[1] = buffer[1];
		rorateAngle=atoi(rAngle);

		//Obtaining isCounterClockWise from buffer
		if(buffer[3]=='C' || buffer[3]=='c')//Counter-Clock Wise
			isCounterClockWise=true;

		//Obtaining riskFactor from buffer
		char rFactor[4];
		for(int i=0 ; i<=3 ; i++)
		{
			rFactor[i]=buffer[i+8];
		}
		int rskFact=atoi(rFactor);
		riskFactor=(float)rskFact/1000.0;

		//Call fn avoidObstacle
		if(rorateAngle>=0 && rorateAngle<=90)
		{
			avoidObs=true;
		}
	}

	//Action : Obstacle Avoidance
	if(avoidObs==true)
	{
		//avoidObstacle(int angle, boolean isCCW, int risk)
		//avoidObstacle(55,false,0.5732); //Test Case
		avoidObstacle(rorateAngle,isCounterClockWise,riskFactor);
	}
```

```
	//Read Encoder
	if (digitalRead(24)==HIGH && isStarted==true && isStopped==false)
	{
		counter++;
		lcd.setCursor(0,0);
		lcd.print(counter,DEC);//Print counter value
	}

	//Read KeyPress Inputs
	int x;
	x = analogRead (0);

	if (x < 100) //Right
	{
		lcd.clear();
		isStarted=true;
		rotateCW();
	}

	else if (x < 200) //Up
	{
		//forward();
		lcd.clear();
		testRun=true;
	}

	else if (x < 400)//Down
	{
		//backward();
		lcd.clear();
		avoidObs=true;
	}

	else if (x < 600)//Left
	{
		//CCW
		lcd.clear();
		isStarted=true;
		rotateCCW();
	}

	else if (x < 800)//Select
	{
		stop();
		isStopped=true;
	}

	if(testRun==true)
	{
		performTestRun();
	}

}

void avoidObstacle(int angle, boolean isCCW, float risk)
{
	isStarted=true;

	//Rotate by 'angle' degrees
	float enc=(136.0/90.0)*(float)angle;
	int encCountR=(int)enc;
	int encCountF=encCountR+(int)(risk*100.0);
```

```
	if(counter<=encCountR && isRotated==false && done==false)
	{
		if(isCCW==true)
			rotateCCW();
		else
			rotateCW();
	}
	if(counter>encCountR && isRotated==false && done==false)
	{
		stop();
		//isStopped=true;
		isRotated=true;
	}

	//Forward once totation done
	if(counter<=encCountF && isRotated==true && done==false)
	{
		backward();
	}
	if(counter>encCountF && isRotated==true && done==false)
	{
		stop();
		done=true;
		isStopped=true;
	}
}

void performTestRun()
{
	//CCW
	//clearLCD();
	isStarted=true;

	//Rotate by 90 degrees
	if(counter<=124 && isRotated==false && done==false)
	{
		rotateCCW();
	}
	if(counter>124 && isRotated==false && done==false)
	{
		stop();
		//isStopped=true;
		isRotated=true;
		//counter=0; //Reset Counter
	}

	//Forward once totation done
	if(counter<=440 && isRotated==true && done==false)
	{
		forward();
	}
	if(counter>440 && isRotated==true && done==false)
	{
		stop();
		done=true;
		isStopped=true;
	}
}
```

```
//Define Low Level H-Bridge Commands
void forward()
{
	//Forward
	digitalWrite(52, HIGH); //Enable A
	digitalWrite(53, HIGH); //Enable A

	digitalWrite(50, HIGH);//Motor A In1
	digitalWrite(51, HIGH);//Motor A In1

	digitalWrite(42, HIGH); //Enable B
	digitalWrite(43, HIGH); //Enable B

	digitalWrite(46, HIGH);//Motor B In3
	digitalWrite(47, HIGH);//Motor B In3
}

void backward()
{
	//Backward
	digitalWrite(52, HIGH); //Enable A
	digitalWrite(53, HIGH); //Enable A

	digitalWrite(48, HIGH);//Motor A In2
	digitalWrite(49, HIGH);//Motor A In2

	digitalWrite(42, HIGH); //Enable B
	digitalWrite(43, HIGH); //Enable B

	digitalWrite(44, HIGH);//Motor B In4
	digitalWrite(45, HIGH);//Motor B In4
}

void rotateCW()
{
	//ClockWise
	digitalWrite(52, HIGH); //Enable A
	digitalWrite(53, HIGH); //Enable A

	digitalWrite(50, HIGH);//Motor A In1
	digitalWrite(51, HIGH);//Motor A In1

	digitalWrite(42, HIGH); //Enable B
	digitalWrite(43, HIGH); //Enable B

	digitalWrite(44, HIGH);//Motor B In4
	digitalWrite(45, HIGH);//Motor B In4
}

void rotateCCW()
{
	//Counter-ClockWise
	digitalWrite(52, HIGH); //Enable A
	digitalWrite(53, HIGH); //Enable A

	digitalWrite(48, HIGH);//Motor A In2
	digitalWrite(49, HIGH);//Motor A In2

	digitalWrite(42, HIGH); //Enable B
	digitalWrite(43, HIGH); //Enable B

	digitalWrite(46, HIGH);//Motor B In3
```

```
        digitalWrite(47, HIGH);//Motor B In3
}

void stop()
{
        digitalWrite(52, LOW); //Enable A
        digitalWrite(53, LOW); //Enable A

        digitalWrite(50, LOW);//Motor A In1
        digitalWrite(51, LOW);//Motor A In1

        digitalWrite(48, LOW);//Motor A In2
        digitalWrite(49, LOW);//Motor A In2

        digitalWrite(42, LOW); //Enable B
        digitalWrite(43, LOW); //Enable B

        digitalWrite(46, LOW);//Motor B In3
        digitalWrite(47, LOW);//Motor B In3

        digitalWrite(44, LOW);//Motor B In4
        digitalWrite(45, LOW);//Motor B In4
}
```

Code Fragment E.1: Control instructions for the robot