MOTION PLANNING AND ASSEMBLY FOR MICROASSEMBLY WORKSTATION

By
ASANTERABI MALIMA

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

SABANCI UNIVERSITY
Spring 2007

MOTION PLANNING AND ASSEMBLY FOR MICROASSEMBLY WORKSTATION

APPROVED BY:

Prof. Dr. Asif ŞABANOVİÇ .....
(Dissertation Advisor)

Associated Prof. Dr. Mahmut Faruk AKŞİT .....

Assistant Prof. Dr. Güllü KIZILTAŞ ŞENDUR .....

Assistant Prof. Dr. Volkan PATOĞLU .....

Assistant Prof. Dr. Hakan ERDOĞAN .....

DATE OF APPROVAL: .....

MOTION PLANNING AND ASSEMBLY FOR MICROASSEMBLY WORKSTATION

Asanterabi MALIMA

EECS, M.Sc. Thesis, 2007

Thesis Supervisor: Prof. Dr. Asif ŞABANOVİÇ

Keywords: Motion planning, microassembly, path planning algorithms, object oriented programming

Abstract

In general, mechatronics systems have no standard operating system that could be used for planning and control when such devices are running. Our goal is to formulate a work platform that can be used as an environment for obtaining precision in the manipulation of micro-entities using micro-scale manipulation tools of our microsystem applications such as our microassembly workstation. The microassembly workstation setup is made up of the manipulation system, vision system, robust control system and manipulation tools. In this thesis we also provide groundwork for motion planning and assembly of the microassembly workstation manipulation system. We implemented the motion planning algorithms which are tested in the virtual workspace environment in order to demonstrate the functionality of the work platform. Firstly, we investigate the performance of the conventional Euclidean distance algorithm, then, artificial potential field algorithm, and finally A* algorithm when implemented on a virtual space. The physical conditions of the microworld hinder the immediate application of the work platform with the motion planning algorithms on the microassembly workstation. We demonstrate our test results of the motion planning algorithms on the virtual workspace and grid window of the work platform. However, due to object oriented programming nature of the work platform, eventually the work platform can be easily interfaced with the microassembly workstation once the problems which limit the micromanipulation and assembly are attended.

MIKROMONTAJ İŞ ISTASYONU İÇİN HAREKET PLANLAMASI VE MONTAJ

Asanterabi MALIMA

EECS, Yüksek Lisans Tezi, 2007

Tez Danışmanı: Prof. Dr. Asif ŞABANOVİÇ

Anahtar Kelimeler: Hareket planlama, mikromontaj, yolu planlama algoritmaları, nesne tabanlı programlama

Özet

Genel olarak mekatronik sistemlerin planlama ve kontrolü için kullanılabilecek standart bir işletim sistemi yoktur. Amacımız, mikrosistem uygulamamızdaki mikro manipülasyon araçlarıyla mikro büyüklüklerin manipülasyonundaki kesinliği elde edecek ortam olarak kullanılacak çalışma düzlemini oluşturmaktır. Mikromontaj iş istasyonunda; manipülasyon sistemi, görüntü sistemi, gürbüz kontrol sistemi, manipülasyon araçları ve yapılacak göreve göre kullanılacak uç takımları için gerekli bağlantı elemanları bulunmaktadır. Bu tezde mikromontaj iş istasyonu manipülasyon sisteminin hareket planlama ve montajı için bir temel oluşturulmuştur. Çalışma düzleminin işlevselliğini göstermek için sanal çalışma alanı ortamında hareket planlama algoritmaları uygulanmıştır. Öncelikle konvansiyonel Euclidean uzaklık olmak üzere, yapay potansiyel alan ve son olarak A* algoritmalarının performansları incelenmiştir. Mikromontaj iş istasyonundaki hareket planlama algoritmalarıyla birlikte çalışma düzeninin hemen uygulanmasını mikro dünyanın engellemesi nedeniyle hareket planlama algoritmalarının testleri ve sonuçları sanal çalışma alanında gösterilmiştir. Fakat, nesne tabanlı programlamanın doğası gereği, hareket planlama algoritmalarını engelleyen problemler modellendiğinde sisteme kolaylıkla dahil edilebilip gerçek uygulamaya geçilebilir.

“To my family and the people I love”

ACKNOWLEDGEMENTS

It is a great pleasure to express my gratitude to all the people who contributed to this work. In particular, I would like to thank my advisor, and mentor Professor Asif Şabanoviç, for his patience, encouragement and confidence. I have very much appreciated the liberty that I was given in doing my work. In addition, he was always there to listen when I knocked on his door.

I wish to thank Dr. Güllü Kızıltaş Şendur for her support, and great advice during my graduate studies at Sabancı University. Also, I wish to thank Dr. Mahmut Faruk Akşit, Dr. Volkan Patoğlu, and Dr. Hakan Erdoğan for their interest in my work and accepting to be my jury members.

Special thanks to Shahzad Khan who not only helped me in brainstorming of several ideas that I worked on during my graduate studies, but also for being a true friend. Many thanks to a great friend, Erhan Demirok whom whenever I saw, it was a sign of a great day. I would like to thank my close friend Erol Özgür; he really has a talent to lighten the candles even when everybody was moody and never felt like talking. Thanks to Erdem Öztürk for being a great leader in the office 1114; he always had the courage to silence us and remind us to get back to work when we seemed to forget our duties. Also, I would like to thank Merve Acer, Meltem Elitaş, and all the other members of Mechatronics Graduate Laboratory for being good companies during my graduate study. Moreover, I would like to express my appreciation to Asha Juma and Esen Aksoy for their wonderful support and word of encouragement all the way. Finally, I would like to express my deepest appreciation to Yousef Jameel Scholarship Foundation of Berlin, Germany for the financial support during my graduate studies.

TABLE OF CONTENTS

LIST OF FIGURES

TABLE OF ABBREVIATIONS

MAW	Microassembly Workstation
OOP	Object Oriented Programming
EDA	Euclidean Distance Algorithm
PFA	Potential Field Algorithm
MMI	Man Machine Interface
HMI	Human Computer Interaction
PZT	Lead Zirconium Titanate
PC	Personal Computer
GUI	Graphical User Interface
DOF	Degree of Freedom
CCD	Charge-Coupled Device

1 INTRODUCTION

1.1 Introduction and Motivation

Microassembly workstation system is an open-structure and reconfigurable system for efficient and reliable assembly of micromachined parts. The structure is used as a research tool for investigation of problems in microassembly. It comprise of manipulation system; which consist of motion stages providing necessary travel range and precision for the realization of assembly tasks, vision system; to visualize the microworld and the determination of the position and orientation of micro components to be assembled, and robust control system and necessary mounts; for the end effectors designed in such a way that manipulation tools can be changed according to the task to be realized.

In this thesis we provide groundwork for motion planning and assembly of the microassembly workstation manipulation system. In the microassembly workstation the vision system provides rich set of features that should allow the manipulation system to move the stages and manipulate the end effectors, and map a larger variety of environment (space). Microassembly workstation also focuses on producing a detailed map of the workspace using image processing algorithms. After detailed information is obtained the manipulation system has to come up with an intelligent motion planning algorithm to achieve a complicated manipulation task depending on the image (scenario) details obtained from vision system.

Humans readily demonstrate the ability to manipulate visually without the supplement of metric information or detailed maps. This ability has inspired the development of motion planning algorithms as a suitable ingredient to the manipulation system so as to reduce human intervention. Such motion planning algorithms have proven to be indispensable for the microassembly workstation manipulation tasks.

For virtual demonstration purposes we prepare a programming window application in which motion planning algorithms are used to demonstrate the functionality of such platform. In this thesis we investigate several motion planning algorithms that can later be implemented in our custom built microassembly workstation. Firstly, we investigate the performance of the conventional Euclidean distance algorithm (EDA) in which manipulation priority is given to the particles that are closest to the destination, followed by use of potential field algorithms to avoid obstacle on the path, and finally, the graph traversing A* (pronounced "A-star") algorithm that finds a path from a given initial node to a given goal (fixed target) node when implemented on a virtual space of our prepared window application.

The software for the manipulation system of the microassembly workstation was developed using object oriented programming environment while the prepared window application contains pull down menus and graphical features to help simplify operation commands of the operator. This thesis covers the description of the environment prepared specifically to be used in microassembly workstation manipulation system with the goal of reducing human intervention in the operation of the system.

1.2 Background and Objectives

In literature, motion planning and control of single and multi-agent robotic systems is a very challenging problem that received a lot of attention in recent years [1]. Challenges in motion planning arise from development of a computationally efficient framework accommodating both the complexity of the environment and the manipulation system control along with the communication constraints, while allowing for a "rich" specification language [2]. In [3], an attempt to discuss planning in micro/nano mechatronics technology such as, precision positioning, micro/nano actuators and microgrippers together with use of macro mechatronics technology in mechanism for intelligent robot control and advanced-user interface can be seen. However, the approach taken in part of this work is to translate the motion planning problem into a 'least cost path' graph problem with an associated cost function for trajectories on the maps. A simple progressive scheme is needed for very large maps giving approximate solutions in reasonable time and memory space [4].

In [5], the motion planning is associated with a pathfinding algorithm so as to determine, first, if it is possible to find a path from the start point to the goal on the map whereby maps may contain impassable barriers, objective is to be able to determine the optimal path. There are many factors that are evaluated to determine the optimal path (defined as the path with the least cost) including the speed in which a path will be traversed. Noticeably, there are many different approaches for defining a path ranging from simple (walk forward until you hit something) to the complex (path finding algorithms with heuristics) [6].

The objective of this work is to present a command language for our Micro-Assembly Workstation (MAW), an idea which was previously introduced in [7] in order to simplify the human machine interaction (HMI). Along with this feature we present common and simple algorithms that are used to demonstrate the functionality of our application. The experimental verification of our work, is more concentrated on micro/nano-manipulation for microsystem application. To achieve this precise manipulation we will formulate a windows application tool in object oriented programming environment which can be used to easily interact with other environment such as signal processing board and image processing tool. Such kind of precise manipulating environment will provide GUI interface which would enable us to change the relative position and relation of entities through direct or indirect human operator control [8]. Given list set of commands on windows application, with motion planning algorithm, we explore the ability of our system to plan path around obstacles or to make choice of the best sequence of arrangements for moving particles to their destinations without any collision with obstacles between them. Interesting feature of our work is to plan motion around obstacles on a map purposely to demonstrate functionality of command language in performing autonomous manipulation of objects in microsystems.

1.3 Thesis Outline

In chapter 2, microassembly workstation (MAW) will be presented. Also, the manipulation system, vision system and control system are presented to demonstrate their role on motion planning. Problems associated with experimentation on the microassembly workstation setup are also presented.

Chapter 3 discusses the features of command language for the MAW. Moreover, information regarding the software for motion planning and control of the Micro-Assembly Workstation (MAW) is covered.

Chapter 4 contains a theoretical discussion of the motion planning algorithm used for micromanipulation. The discussion about the proximity (Euclidean distance) based algorithm for manipulation of particles followed by artificial potential field approach, and the graph traversing algorithm A * is presented.

In chapter 5, test for single command operations given on the window application are shown followed by experiments and results conducted by using different motion planning algorithms is covered.

Chapter 6 concludes the thesis by pointing out the achievements and giving future motivations in this research project.

2 MICROASSEMBLY WORKSTATION

Many problems arise through the miniaturization process because of the scaling effects, manufacturing techniques and, as might be expected, assembly. The production of microsystems integrated with much functionality, many components made of different materials require flexible, modular, accurate mechanisms, which can finely pick, orient, move and release different types of objects at the right place. In the presence of microparts assembly is a key issue in the formation of a product since different functions require different materials within a product. In addition, microassembly is the key for the development, modification and maintenance of hybrid microproducts.

In comparing relations between the production rate and the size of the parts to be assembled, as the assembled parts become smaller and smaller the production rate rapidly falls thus opening a field of research in the micrometer size components assembly. Due to these facts, the operator has no direct access to the micrometer size components and since direct human handling of those components is not possible, the need for automated micromanipulation and microassembly is crucial (Figure 2-1).

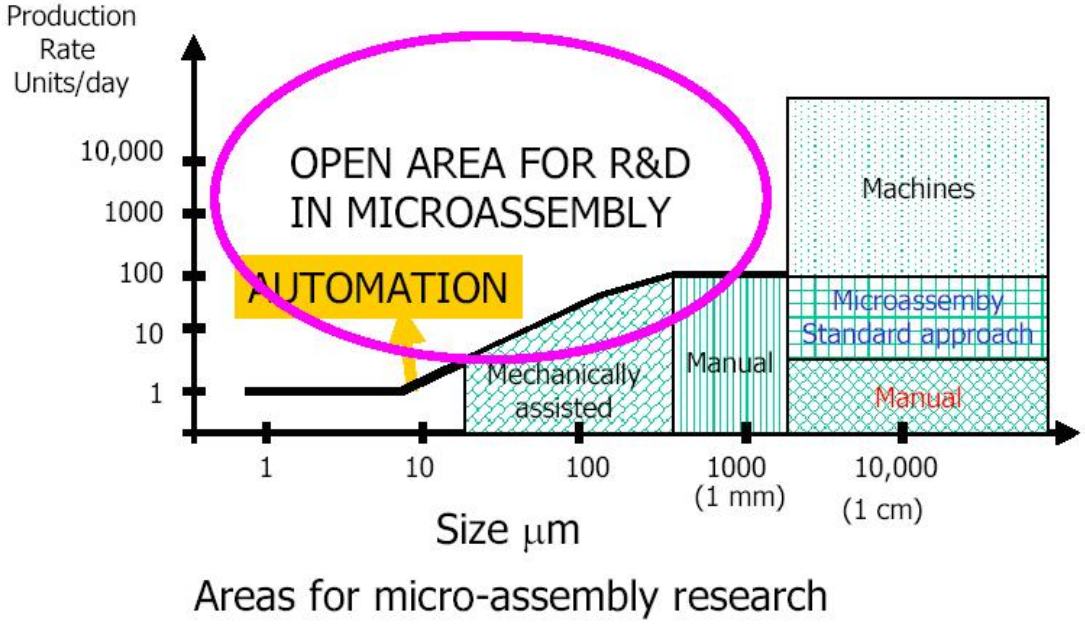


Figure 2-1 – Interrelations between areas of microassembly

2.1 Types of Microassembly

According to the technique used, microassembly can be classified as shown in Figure 2-2. In additional to these techniques stated, there are some special approaches but will not be presented here. What we present in this thesis lies along the automated assembly motion planning, however in the following subsections we provide some background information of what is covered in other braches of microassembly as well.

2.1.1 Manual microassembly

Manual microassembly is the realization of assembly tasks by specially trained by human operators where they need to perform high-precision hand motions. For the manual microassembly there is a great need for visual aid by magnifying glasses or microscopes under which the operators carry out the assembly tasks by using special tools like tweezers. As a result of the small dimensions of the parts to be assembled, forces that are to be applied in order not to give damage to the parts are restricted. Manual assembly may result in varying product quality since the control of such small forces is hard and may lead to deformations of the parts. In addition, with the miniaturization increases, the mounting tolerances are becoming smaller and smaller which makes the manual assembly harder since the capabilities of the human becomes inadequate in that scale.

2.1.2 Tele – operated Microassembly

In tele-operated microassembly, motions of the human operator are transferred into the actuators by means of a man-machine interface (MMI). MMI having more number of degrees of freedom enables the control of the motion with the same number of DOF in the target space. Since the forces and the precision of motion necessary for realizing an assembly operation in the microworld should be too small and the human capabilities become inadequate for such motions with the increasing miniaturization, tele-operation makes it feasible to perform the operations remotely, not in contact with the environment. It allows the realization of the manipulation operations precisely by scaling the forces and the motions of the microworld to the macroworld and feeding

back to the human operator. So that the operator can perform the operations similarly as in the macroworld and the task is realized by the micromanipulator system in the microworld. During those operations, the operator is guided through visual feedback and possibly with force feedback.

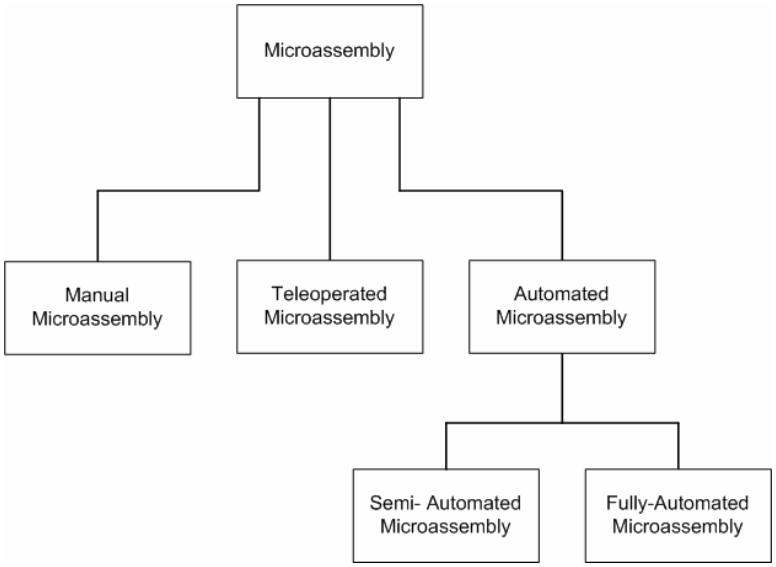


Figure 2-2– Classification of Microassembly Techniques

2.1.3 Automated Microassembly

Main issues related to the microassembly are the component handling, precision and the final product quality. Component handling difficulties, required submicron precision necessary for the quality of the product and the limits of human capabilities brings the necessity for an automated microassembly operation. Besides guaranteeing the required precision and repeatability, automation is necessary in the microassembly also for economic reasons since automation increases the quality of the product and the production rate which will reduce the cost.

Automatic microassembly can be divided into two categories

- Semi-automated Microassembly
- Fully-automated Microassembly

In semi-automatic microassembly, operator intervention is permitted but only to some extent. The operator can define some parameters for the operation such as the pick-place positions or the assembly order of the components. The rest of the operation is executed automatically. In fully-automated microassembly, all the tasks and the

parameters are predefined. With the aid of sensory feedbacks such as visual feedback, force sensors, etc. the assembly task is realized automatically. In this regard, what we add in this thesis is the ability to reduce the human intervention in the semi-autonomous manipulation so as to improve our system towards fully autonomous operations. Further discussions on the state of the art and the existing microassembly systems can be found in [16].

2.1.4 The Subject of this Research

In this thesis motion planning algorithms are implemented to demonstrate the ability to reduce human intervention on the microassembly workstation by allowing several commands to take place at once to achieve a certain task. The operator just chooses the motion planning algorithm to be implemented and in one of the algorithm a specific particle to be manipulated is selected from a crowd, then using the algorithm the path which can be used to intelligently manipulate the particle towards its defined destination is shown. Even though in semi-automated microassembly the operator intervention to the assembly operation is limited, the motion planning algorithm helps to reduce the human intervention further. The tasks are pre-programmed and the operator simply defines some of the parameters necessary for the assembly. The destination of the manipulated piece is usually given as a pattern depending on the task to be fulfilled. The rest of the operation is managed automatically. Semi-automated system architecture is shown in Figure 2-3; the figure demonstrates the procedure which is: after the handling of the initial steps by the operator, relative distance data calculated by the vision system is fed to the motion system while sensory feedback (position data) from the motion stages is sent back.

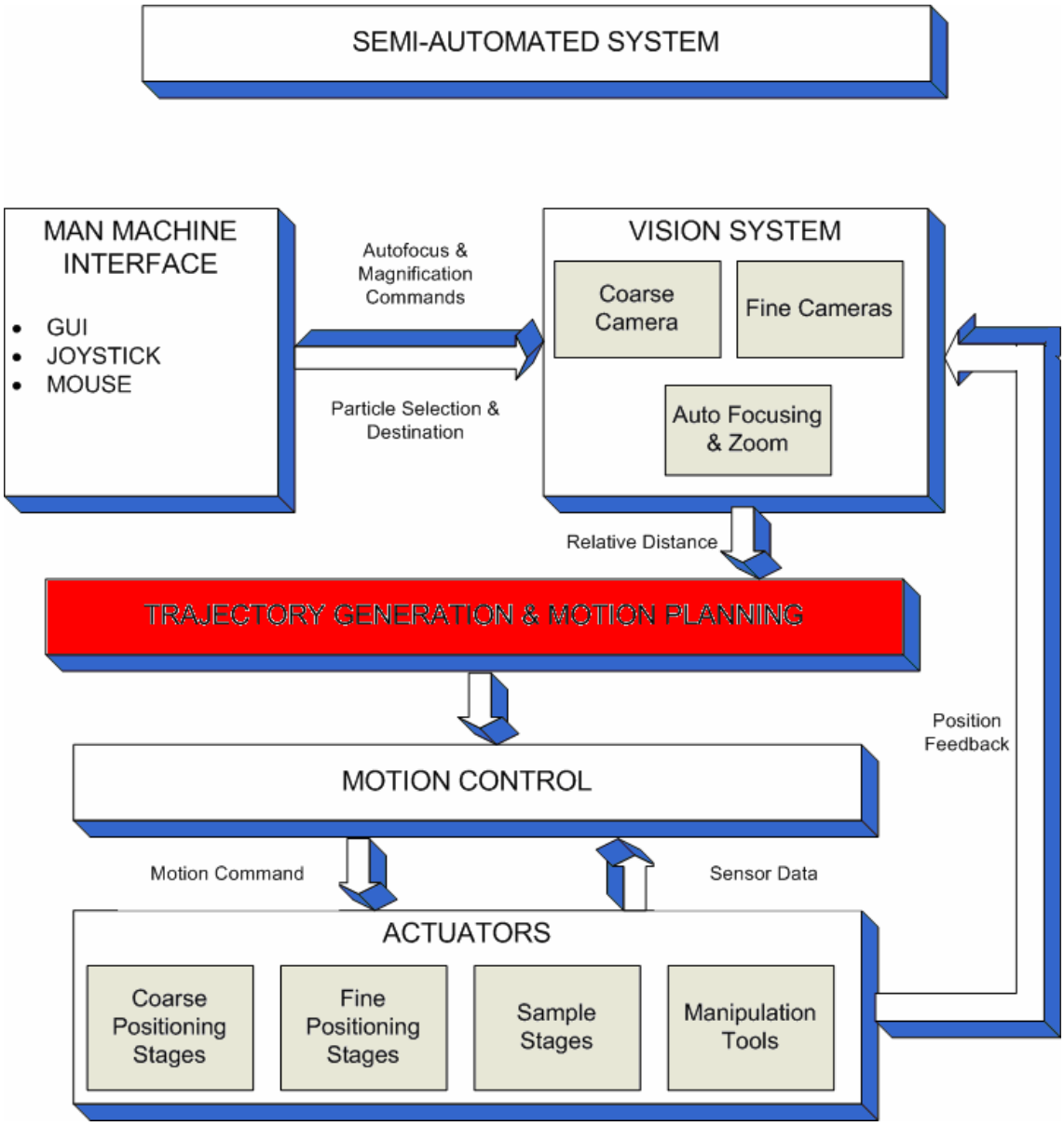


Figure 2-3 – Trajectory Generation and Motion Planning in Semi-Automated System Architecture

Figure 2-3 also shows the concentration of this thesis work as an intermediate between the vision system and motion control system in the microassembly workstation. Note that in semi-autonomous manipulation, the system initialization includes the selection of the particle and the desired destination point. Later, a line connecting the center point of the selected particle and the destination particle is constructed. Then, step motion value for pushing the particle and the selected point are calculated. Finally, the tip of the probe is moved in steps towards the destination point in order to achieve semi-automated manipulation. The steps are repeated until the center of the selected particle coincides with the desired point [16].

The most feasible trajectory for a particle to its target position by pushing is simply a line denoting the closest path to the destination. In that context, the operator should choose the suitable part to be pushed and the destination point considering the issue that the semi-automated assembly procedure does not include motion planning so that there exists nothing as an obstacle between the particle and the target point. In this thesis work a similar feature is added as Euclidean distance planning algorithm; in which the path planning of the particles to be manipulated are obtained depending on the particles proximity to their destination points.

2.2 System of Microparts Manipulation Workstation

The overall structure of the workstation is presented in Figure 2-4. The system has the following main components:

1. Manipulation System consists of the gripper manipulator and the sample stages, providing the motion necessary for the manipulation operations.
    a. Gripper manipulator consists of coarse and fine positioning stages providing necessary travel range and the accuracy for the manipulation stages mounted on the fine positioning stages.
    b. Sample Stages provide the usage of the substrate surface more effectively by moving the different regions of the substrate into field of view.
2. Control System is the main control unit of the system consisting of a PC and a controller board embedded into the PC.
3. Vision System can be examined in two parts;
    a. Optical System consisting of a stereoscopic optical microscope, CCD cameras mounted on the microscope and an autofocusing control unit. The automatic zoom and the autofocusing motion units are mounted on the microscope and the control is realized from the vision computer.
    b. Illumination System is configured to provide backlighting by means of a mirror system using a fiber illuminator as the light source.
4. End Effectors and necessary fixtures are used interchangeably in the system. Microgrippers and probes can be the matter of choice and necessary fixtures are designed to be easily integrated to the system.

5. The whole system is placed onto an actively controlled damping table in order to get rid of environmental vibrations.

The whole system configuration is shown in Figure 2-4.

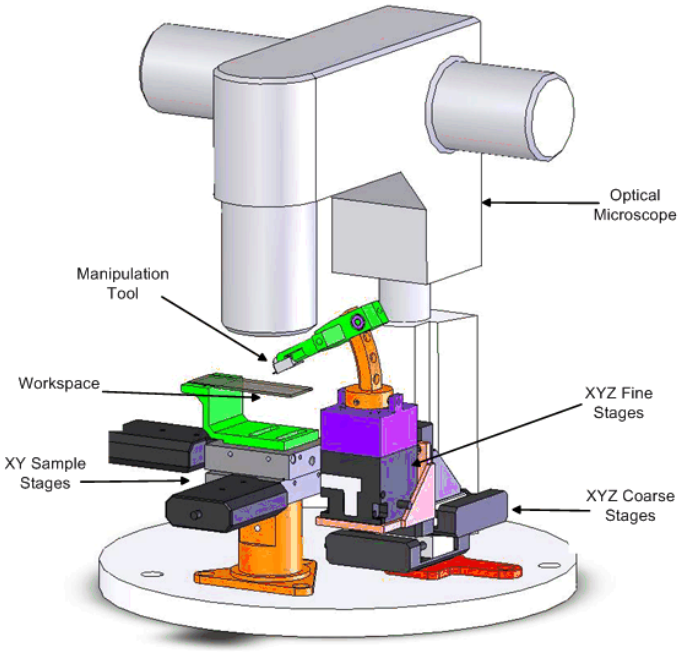


Figure 2-4– Microassembly Workstation

2.2.1 Micromanipulation System

Manipulation system should allow the positioning of the end effector with very high accuracy. According to the complexity of the manipulation tasks, the manipulator system should have necessary number of the degrees of freedom. With translational degrees of freedom in x, y and z coordinates, system will be capable of executing simple pick and place tasks and some 2D tasks such as pushing, pulling type of assembly tasks. To be able to implement more complex assembly tasks and to provide the system to adapt to any kind of assembly process, a rotational degree of freedom about the z axis was added to the manipulator system.

2.2.2 Control System

The micro assembly workstation will serve the purpose of manipulating micron and sub micron sized modules, and assembling them to build complex but small sized

machines or modules thereof. This section focuses on the design methods for the implementation of control system aspects of the microassembly workstation. The system can be made up of the following modules:

- Positioning stages with different type of actuation mechanisms
- Manipulation tools (e.g. microgrippers)
- Vision system with cameras and an optical microscope which has actuated focusing and zooming features
- Man-machine interface consisting of devices that relay concise information to the user including a haptic device for information exchange.
- Main control computer as the computing engine running in real-time, processing the data coming from various parts of the system and producing reference values for the modules.
- Real-time communication network to provide the connectivity between the modules.
- Non-real time communication network to communicate between parts of the system where occasional late arrival of data can be tolerated. (e.g. connection between the main control computer and the haptic device.)

The modules of the system listed above is configured in such a way that they can operate independently of each other. In order to provide the functioning of the whole system, all modules must be coordinated. For the coordination of the modules, there are several ways to implement the data processing and the control system.

Main considerations of the system supervision and control are related to the motion control and image processing. In order to achieve high accuracies for the motion, control method needs to be considered carefully. Motion control is the basis for the microassembly workstation since the precision and accuracy of the assembly tasks mostly depend on the control performance. Vision system supplies the external sensory feedback for the motion control unit. By using some image processing techniques, position of parts to be manipulated and the manipulation tools with respect to each other which are necessary for the implementation of automated assembly tasks are extracted. As the motion units operate according to the feedback supplied by visual feedback, vision system must be very precise for the sake of realization of the assembly tasks. For the automated tasks vision system gains significant importance as it gives the reference for motion according to the extracted position information.

2.2.3 Vision System

In the micro domain, position and orientation of the parts to be assembled with respect to each other and the manipulation tool can be determined only by using a vision system. An effective vision system is necessary to visualize the micro domain for recognizing the geometries and position of 3D microparts, path generation for the micromanipulator and providing the necessary position feedback. Application of different vision technologies depends on the size of the parts to be manipulated. The assembly task for the microassembly workstation is defined in the micrometer domain, so that optical microscope and contact manipulation are suitable solutions for the vision system.

Configuration of the vision system is shown in Figure 2-5.

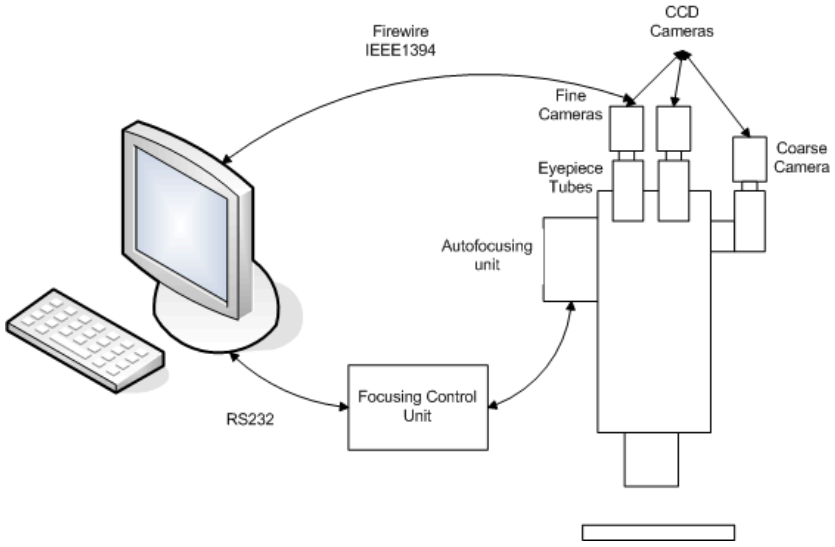


Figure 2-5 – Vision System

2.2.4 Manipulation Tools

One of the most essential components of microassembly systems is the manipulation tools determining the manipulation capabilities of the whole system. Manipulation is the key operation for a microassembly workstation as the assembly can be realized by means of manipulation.

Unfortunately, there is not a wide range of choice of micromanipulation tools in the market as the microtechnology is still an immature research area all over the world.

A few microgrippers are available in the market and suitable ones for our purposes are selected to be Zyvex Nanoeffector® Microgrippers and Nascatec Nanogrippers.

Microgrippers make it possible to realize 3D microassembly operations. With the available range of microgripper sizes it is possible to manipulate different objects with different sizes and features. More details on the experimental setup are available in [16].

2.3 Microassembly Workstation Operation

This section discusses the current features of the microword which limit the immediate application of the motion planning algorithms on the microassembly workstation.

Differences between the assembly in macro and microworld forms the basic requirements for the design of a microassembly workstation which can be defined as speed, repetitiveness and reliability. As the system is defined as an open architecture and reconfigurable system – both hardware and software – the system is designed in such a way that it can adapt easily to different applications. Therefore, flexibility comes out as another design requirement for the workstation.

Noticeably, the microassembly workstation should have the following requirements to operate with a good performance;

- For the precise handling of the micro parts, a manipulation system with high accuracy
- Robust control system design is of great interest in the microassembly. Modeling and control, especially vision assisted control, become more critical in microassembly as the accuracy requirements increase and the size of parts decreases.
- For the determination of the position and orientation of microparts, a sophisticated measuring system is needed such as an optical microscope for guidance in assembly of microparts.
- Robust control system design is of great interest in the microassembly. Modeling and control especially vision assisted control, become more critical in microassembly as the accuracy requirements increase and the size of parts decreases (That's why in our previous work we investigated the high precision controller for motion control) [19].

2.3.1 Physics of Micromanipulation

With the miniaturization of the objects to be handled to micron sizes, attractive forces, such as van der Waals force, surface tension forces and electrostatic force between microobjects and the manipulator become dominant over the gravitational and inertial forces.

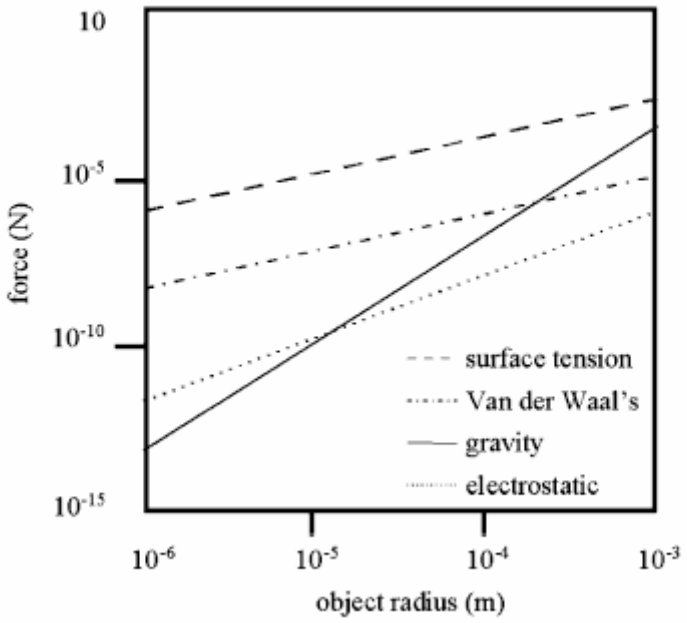


Figure 2-6 – Comparison of gravitational and adhesive forces [11]

These attractive forces depend on environmental conditions, such as humidity, temperature, surface condition, material, etc. For the manipulation of the microobjects, the physics in the microworld should be carefully considered. Thermal, optical, electrical, and magnetic effects will change or become dominant when the objects are miniaturized (Figure 2-6).

2.3.2 Discussion on Mechanics of Micromanipulation

As described in the previous subsections, the action of micromanipulation is very complex and possessing several problems because of the adhesive forces, complicating the pick and place tasks. Requirements for the microgrippers differ from the conventional grippers not only for the size of the particles to be manipulated but also for the required gripping forces. As a result of that for the design of micromanipulation

tools, it is not possible to just miniaturize the existing manipulation tools in the macro domain. According to the tasks to be realized and parts to be manipulated in the micro domain, a specific manipulation tool is necessary which can perform not only gripping but also releasing of the particles. Investigation on properties of different kinds of grippers is covered in [16].

Furthermore, minimizing the contact surface is necessary to reduce the adhesive forces dominant in the microworld. In order to avoid the sticking effects, contact surface between the probe and the particle should be as small as possible [10], [12]. The optimum configuration for the pushing operation using sharp tungsten probes is depicted in Figure 2-7. In this configuration, the probe tip and the particle is in minimum contact providing an effective manipulation.

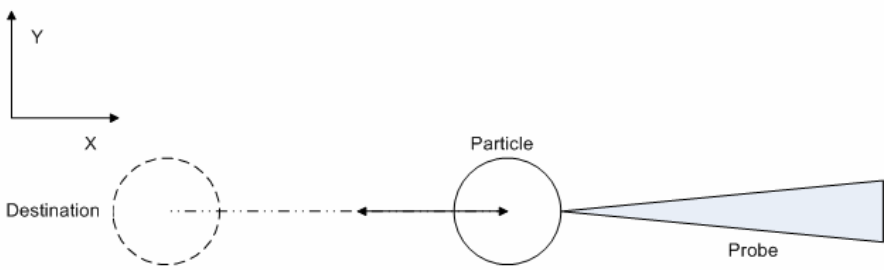


Figure 2-7 – Optimum pushing configuration [16]

During the pushing operation, the probe tip should point the center of the particle since the particle can roll around itself failing the pushing operation. Furthermore, if the distance determined for pushing operation is too much, the particle can also roll around itself due to imperfections of the base surface and the particle shape, not touching the surface at one point, etc. Therefore, the operator should be trained to realize experiments by using the probe as the manipulator. Also, intelligence in the discretization of the path in motion planning is crucial to avoid the particles from rolling around during the pushing operations. Moreover, the addition of the rotation axis to MAW improves the ability to manipulate microparticles in the workspace, see Figure 2-8.

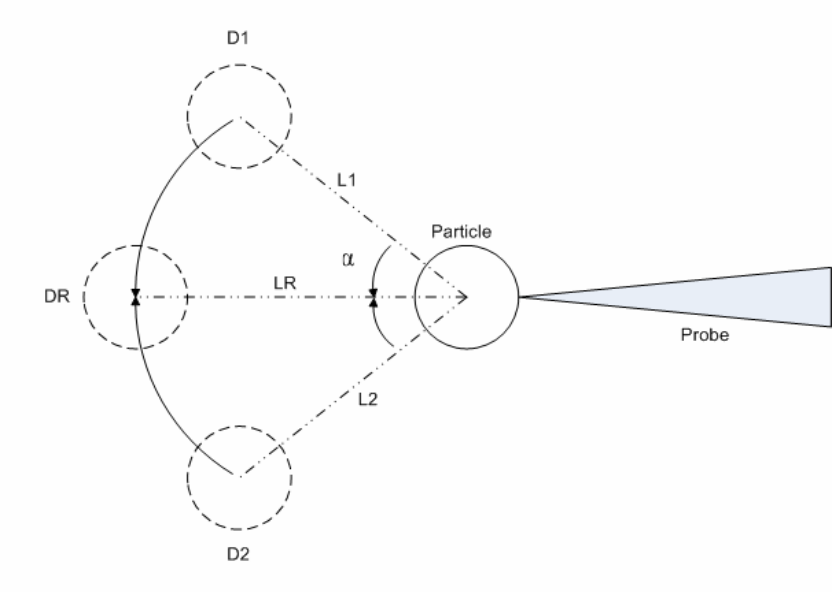


Figure 2-8 – Addition of a Rotational Axis

Considering the experiments realized in the microassembly workstation, results are promising when precision, accuracy, reliability and repeatability issues are concerned. However, when the discussed phenomena are dealt with, the system can operate effectively as a microassembly workstation for automated assembly tasks. Eventually, accuracy requirements for these operations are very high; therefore, precision and repeatability of such assembly systems must be in the micron to nanometer range for automatic assembly of millimeter and micron structures. In the next chapter, features of MAW software that guarantee precision and repeatability in the assembly processes will be discussed. In the next chapter, features of MAW software which guarantee precision and repeatability in assembly processes are discussed.

3 SOFTWARE FOR MICROASSEMBLY WORKSTATION

3.1 Introduction

This chapter is the introduction of the software to be implemented on the microassembly workstation. In order to setup an open-architecture and reconfigurable microassembly workstation with the necessary specifications, issues related to the manipulation language of the system components will be discussed.

In the following sections, the whole system will be examined in detail as software architecture, structure of the software and operation of the software. Finally, software configuration to improve the system performance will be presented.

As discussed earlier, our motivation for this work is to create a language in an object oriented fashion in which a user would give some commands and the system would respond accordingly to execute the tasks. In attacking the problem, the motion plan is obtained through model checking, and results in the form of graphical representation of the provided workspace. A brief discussion on the structure and architecture of the software is covered in this section; however, the following section will also cover discussion of how the software architecture results to computational complexity. Manipulation tasks are specified by identifying a target object to be moved and its new location.

The work of thesis involved preparation of the window application with its features shown in the Figure 3-1. The commands are classified into groups in terms of operations they handle in motion planning and assembly of the microassembly workstation. Features of the Main window application prepared include:

- *Main Menu Strip*: Carries the implemented items which are located on the *Main Menu Item* bars

- *Graph Traversing Control Commands*: Are commands used to control graph traversing A* algorithm
- *Graphics Menu*: Contains the menus for demonstration purposes of the A* motion planning algorithm
- *Stand and End*: Are initial and final positions of the motion planning routine
- *Path Finding Time*: Is the time completed in the process of path planning using the A* algorithm
- *Action Control Commands*: Contains all the actions implemented for the demonstration of motion planning in the virtual space and "Grid Search Window"

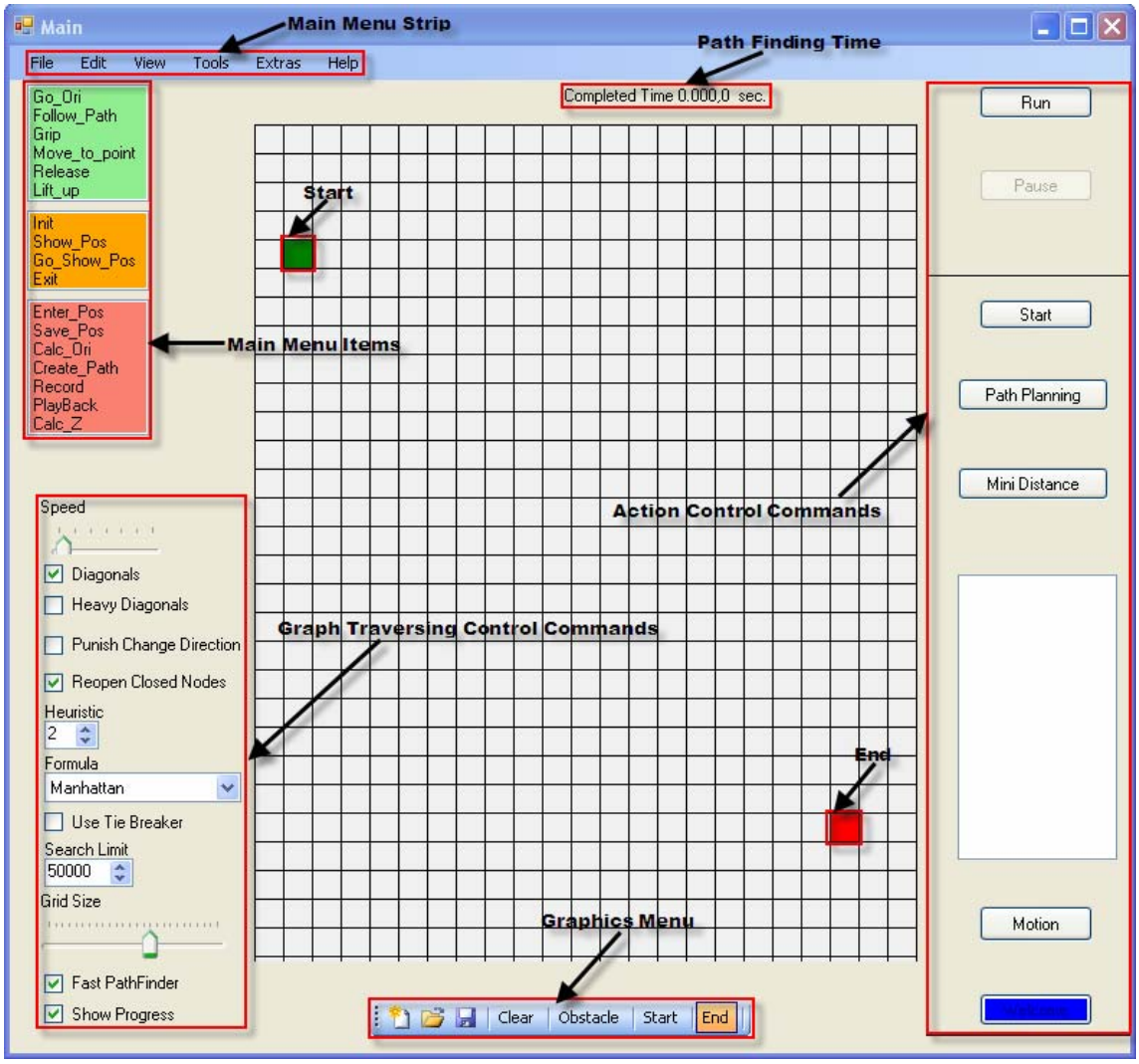


Figure 3-1 – The window application used to control operation of the MAW

3.2 Software Architecture

In motion planning, given the high cost of computing optimal plans using classical search algorithms, we focus on divide-and-conquer techniques and on the efficient reuse of existing plans.

We prepared the list set of commands presented in the *divide-and-conquer approach* for the identification of control modes and their combination can lead to major task in the manipulation process with emphasis on precision, accuracy, reliability and repeatability. To achieve this we prepared classes for the whole system. Then, all the system is divided into several categories with particles, stages, microscope and manipulation tools as **objects** which are represented by the defined classes. e.g. For the real time experiment in MAW we have polysterene ball images represented with the following **properties**: circular objects (in 2D), with a given radius r, and with x and y coordinates of the center. Collection of these mentioned properties can be used to represent **methods** such as rotate, push, record coordinates of particular particle. Figure 3-2 below illustrates how the manipulation of a particle takes place without colliding with other particles (obstacles) along its way towards destination.

Figure 3-2 – Manipulation of particles on the virtual space

In early days, structured programming is the type of programming which was commonly used. Now, before we introduce the features of object-oriented programming reviewing the dominating programming approach prior to object-oriented programming is very crucial.

Structured programming relies on use of high-level control structures instead of low-level jumping. Structured programming is also loosely coupled with top-down programming and program development by stepwise refinement. A well-structured program should devote a single procedure to the solution of a single problem. The splitting of problems in sub-problems should be reflected by breaking down a single procedure into a number of procedures. Notice that only few programmers are radical with respect to top-down structured programming. In the practical world it is probably much more typical to start somewhere in the middle, and then both work towards the top and towards the bottom. Even though structured programming has some weaknesses in real-time applications this does not mean structured programming is "the wrong way" to write programs. Similarly, object-oriented programming is not necessarily "the right way". However, object-oriented programming (OOP) is an alternative program development technique that often tends to be better if we deal with large programs and if we care for program reusability.

When we write a "traditional" structured program it is most often the case that we have a single application in mind. This may also be the case when we write an object-oriented program. But with object-oriented programming it is more common - side by side with the development of the application - also to focus on development of program pieces that can be used and reused in different contexts.

The next observation deals with "stable structures". What is most stable: the overall structure of program control, or the overall program data structure? The former relates to control structures and procedural structures. The latter relates to data types and classes (in the sense to be discussed in the following section). It is often argued that the overall program data structure changes less frequently than the overall program control structure. Therefore, it is probably better to base the program structure on decomposition of data types than on procedural decomposition.

Eventually, the above discussion brings us to our interests towards programming "the object-oriented way". Below we list some of the most important ideas that we must care about when we make the transition from structured programming to object-oriented programming. First of all, the gap between the problem and the level of the machine is

filled in the bottom up style. Also, data is used as the basic building blocks, however, some people argue that data, and relations between data, are *more stable* than the actions on data. Then, data is bundled with the natural operations on data: which is one of the fundamental ideas of *abstract datatypes*, and also consolidate the programming constructs (structs/records) for encapsulation of data. Object-oriented programming concentrate on the objects which should be administrated or handled by the program: it make use of existing theories of phenomena and concepts, and in particular it provides features which help to form new concepts from existing concepts. Object-oriented programming make use of programming style that allows us to collapse the programming of very similar objects. The following section clarifies the structure of the software used for motion planning of the MAW.

3.3 Structure of the Software

The structure of the MAW software as described in [7] is in Backus-Naur Form; Backus-Naur notation (more commonly known as BNF or Backus-Naur Form) is a formal mathematical way to describe a language, which was developed by John Backus (and possibly Peter Naur as well) to describe the syntax of the Algol 60 programming language. In [7] the structure of the software had the following features:

Define Commands are useful when user wants to set a certain condition for MAW or change some behaviors in MAW.

Enter_Pos ::=**Enter_Pos** "("name "," (x-value "," y-value)")"

Use ::=**Save_Pos** "("name "," object")"

Use[*1] ::=**Calc_ORI** "("object1 "," object2 "," variable ")"

Use ::=**Create_Path** "("name "," {variable}|{x-value "," y-value}")"

Record ::=**Record** "("filename | close ["," deviceID")"

Playback ::=**Playback** "("filename")"

Device_Ori ::=**Device_Ori**"(" device ID "," setting")"

[*1] Exception: Argument '**variable**' in Command **Calc_ORI**( ) use variable 'STATUS' field as a Flag (**Flag=1**, Rotation is necessary, angle of rotation θ_z is given in 'VALUE' field; **Flag=0**, No rotation, Value field has 0 (θ_z=0)).

Movement Commands are commands which initiate movement in MAW.

Go_ORI ::=**Go_ORI**"("move_mode "," variable ")"

Follow_Path ::=**Follow_Path**"("move_mode "," name ")"

MovetoPoint ::=**Move_to_Point**"("move_mode "," (x-value ","y-value)|variable")"

Grip ::= **Grip**"("object")"

Release ::= **Release**"("object")"

Lift_UP ::= **Lift_UP**"("object ["","z-value]")"

Action Commands are commands which handle sensors and other equipment in MAW.

Init ::=**Init**"("[address]" , "[{address}]")"

GoShowPos ::=**Go_Show_Pos** "("move_mode", "view_mode", "object")"

Show_Pos ::= **Show_Pos** "("sensorID | view_mode "," object [","forever]")"

Then, there are *Control Flow Commands* which are commands used to control execution of other commands.

If ::=**If** "("expression")"{commands} **endif**

While loop ::=**While** "(" expression")"{commands} **endwhile**

Break ::=**Break**["("name")"]

Stop ::=**Stop**

Cont ::=**Cont**

Clear ::=**Clear** "("variable")"

Wait ::=**Wait** "("number |variable")"

***Note*:** Commands and parameters used here are expressed using the BNF-notation (Backus Naur (BNF) Form)

After introducing the concept of object oriented programming in the previous section, the following is a brief content of the objects used in the software which ensure the software reusability.

- "*Manipulation Point*": Is an **object** which contain two points with each point having x and y coordinates of the respective point; e.g. first point could be a representation of origin point while second point could be a representation of the destination point. **Methods** used to calculate the distance, slope, angle, destination path, destination force and obstacle force between the two points utilize the **properties** of this object.

- "*Particle Store*": Is an **object** which carries all information of the particles used in the virtual space manipulation. Method such as minimum distance between all the particles in store and the destination point is handled using this **object** class.
- "*Path Finder Node*": Is an **object** which contains information of the nodes such as: start, open, close, current, path and end. These nodes information can be used to define **methods** such "Find Path" using defined heuristics.
- "*Point*": Is an **object** which carries information of x and y coordinates generally used as an object of another objects such as "*Manipulation Point*", "*Path Finder*", etc.
- "*File Stream*": Is an **object** which hides all information of the nodes present in the GUI interface grid space, so that the information can be used for path planning routines, or saved in the memory and retrieved for future use.
- "*Form*": Is an **object** window or dialog box used for graphics demonstration purposes; e.g. "Virtual Workspace" and "Windows Media" form. Form is one on the features of C#.NET programming environment.
- "*Priority Queue*": Is an **object** which keep list of nodes information while the A* algorithm is searching for the optimal path in the grid space.
- "Timer": Is an object used to calculate simulation time in seconds

Based on the structure presented in [7] and classes of object defined above, the software for the motion planning of the MAW window application was divided into three categories which are:

- *Movement Commands*: Motion commands such as "Follow Path", "Move to Point", "Lift Up", "Rotate(change orientation)" and "Return to Origin" are categorized in this part;
- *Action Commands*: Commands to "Initialize Manipulation", "Show Position of the Moving Part" and "Exit Manipulation" actions are presented; and
- Definition *Commands*: List of basic definition manipulation of particles such as "Enter Position", "Save Position", "Select Motion Planning Algorithm", "Create Path", "Save Movie", "Play Movie", etc.

In order to prepare the main window application some features were taken from [15], these include:

- *Grid Size*: This parameter just affects the front-end. It can change the grid size, where reducing the grid size gives a chance to create a bigger test but will take longer to render.
- *Fast PathFinder*: When unchecked, the implementation used is the algorithm as it usually appears in Path Finder [15]. When checked, it will use path finder implementation which requires more memory, but it is about 300 to 1500 times faster depending on the map complexity.
- *Speed*: This is the rendering speed; reducing speed permits detailed examination of how the algorithm opens and closes the nodes in real-time while the PathFinder operates.
- *Diagonals*: Is set to allow the A* algorithm to process path searching in 8 directions instead of 4; including the diagonals of the grid.
- *Reopen Closed Nodes*: Is the command that allows A* algorithm to reopen nodes that were already closed when the cost is less than the previous value. If reopen nodes is allowed it will produce a better and smoother path, but it will take more time.
- *Formula*: Is the equation used to calculate the heuristic. Different formulas will give different results: some will be faster, others slower and the end may vary.
- *Punish Change Direction*: Is the command that allows every time the A* algorithm path finder changes direction the cost decreases. The end result is that if the path is found it will be comparatively smooth without too many direction changes, thus looking more natural. The downside is that it will take more time because it must research for extra nodes.
- *Show Progress*: This permits observation of the algorithm as it operates in real-time. If this box is checked, the completion time will be the calculation time plus the rendering time.
- *Tie Breaker*: In A* path planning algorithm sometimes it encounters a phenomena in which there are many possible choices for the same cost and destination. The tie breaker setting tells the algorithm that when it has multiple choices to research, instead it should keep going. As it goes, changing costs can be used in a second formula to determine the “best guess” to follow.
- *Completed Time*: Is the time the algorithm takes to calculate the path from the start to end point. To know the true value, uncheck “Show Progress.”

- *Run/Continue and Pause* Are action control command buttons use to run and control the A* motion planning algorithm. The Pause command is used to pause the graph traversing process while the search for the path continues.

All these functions are featured in the window application main form as seen in Figure 4-2.

3.4 Operation of the Software

Manipulation Process:

- "Initialize Manipulation – Action command" initializes the manipulation process, in which the manipulators move to the origin (home) position which is known,
- "Select Algorithm – Definition command" enables user to select the appropriate algorithm (discussed in the next chapter) to be used in the particular task.
- "Save Movie" is the command used to save the manipulation process as avi.file so that it can be replayed using
- "Play Movie – Definition command" later for further analysis.
- "Return to Origin – Movement command" is the command for return all stages and manipulators to the start (system initializing position) and
- "Exit Manipulation – Action command" is used for stoping the manipulation and switching-off the application.

By default, program can also be set in such a way that once the object has been placed at the target (destination) location, the manipulator can return to its rest position (system home position). In addition, hierrachies are assigned in the execution of the commands to ensure that priorities are given to a set list of commands e.g. Action commands.

Our project challenges involved writing the codes for the commands which run multiple systems at once. This feature, however may lead to several outcomes such as delay in debugging due to computational complexities resulting from handling all the data coming from multiple sensors (camera, motors and manipulators) and sending the appropriate commands to the motors and the manipulating tools. Implementation of such commands can be observed in the motion planning algorithms in the next chapter.

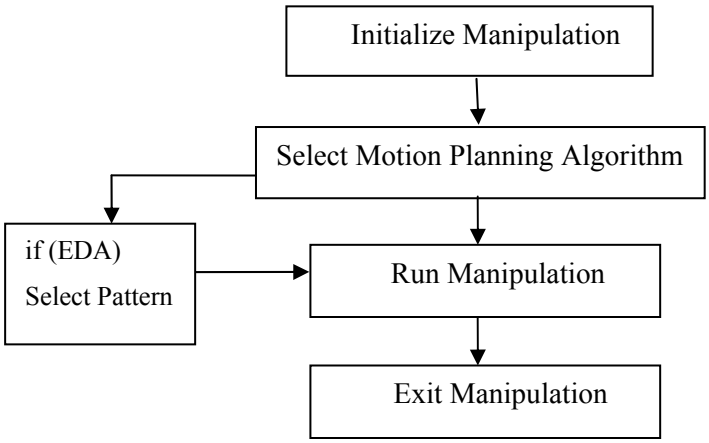


Figure 3-3 – The manipulation procedure using the software

The window application can be put into two categories, the first one involves commands to perform single commands at once, usually commands which can be applied in the assembly process and the second one includes collection of systematic list of commands given when system is required to perform certain action. The manner in which the motion planning algorithms is set to perform manipulation of micro-particles fits into the second category of the window application.

3.5 Conclusion and Discussion

In this chapter, detailed structure and implementation of the software part of the microassembly workstation was discussed by means of explaining each subsystem separately. We have demonstrated how the tasks are specified in a high level language and have the manipulators and stages automatically convert these specifications into a set of low level primitives to accomplish such tasks. Functionality of the software prepared in the object oriented manner is demonstrated in the following chapter. The common advantage of preparing the MAW software using object oriented programming makes the software to be easily developed and easily reusable in other mechatronics system operating in same manner.

4 MOTION PLANNING ALGORITHMS

4.1 Introduction

In this chapter, motion planning algorithms used for manipulation processes of the microassembly workstation will be presented. Detailed discussions of why and on which scenarios an operator would prefer to apply one algorithm instead of the other is also presented. Then, mathematical calculations involved in the motion planning algorithms and results expected from virtual space implementation are described.

Consider a workspace with randomly distributed microparts, our objective is to use motion planning intelligent algorithms to formulate patterns of particles in defined locations on the workspace. In motion planning procedure a good understanding of the physical feature of our manipulation system is very crucial before the user issues commands. Then, after all constraints have been put into consideration the application of this technology in assembly of micro components and structures into microsystems can be fulfilled. Common application of this planned manipulation is on construction of useful 2D microstructure, e.g in fabricating mold templates in micro/nanoprinting [3]. With few exceptions, most of the works in this area focus on either the complexity of the environment or manipulator dynamics (while assuming the environmental features are trivial). Previous study show communication architectures in multi-actuator systems focusing on proving that certain local interactions give rise to interesting global behaviors. However, the inverse problem of generating local rules from non-trivial high level specifications of the group is still not understood. In most of the existing works, the motion planning problem is simply specified as "go from A to B". It has been discussed by several authors [2] that this kind of observation is either too explicit, or simply does not capture the nature of the task, which might require logical (e.g., "visit either A or B") and/or temporal operators ("reach A and then B infinitely often").

Technique for performing motion planning with obstacles range in complexity from simple behavior-based approach to complex global path-planning schemes. The simplest approaches are reflexive in nature. Some examples include: turning left if an obstacle is detected on the right and turning right if an obstacle is detected on the left, walk forward until you hit something. These approaches tend to be very robust and adaptive to unstructured environments, but also tend to be inefficient and non-optimal. Motion planning can find optimal paths through complex environments but also tends to be brittle and to not scale well to large environments [13].

For demonstration purpose of the motion planning algorithms implemented in this work we will use two space: first one is a "grid search window" in the middle of the main form of our widow application usually used for demonstrations of the A* algorithm, and second one is the "virtual space" form which is activated when either Euclidean distance or artificial potential field algorithm is activated.

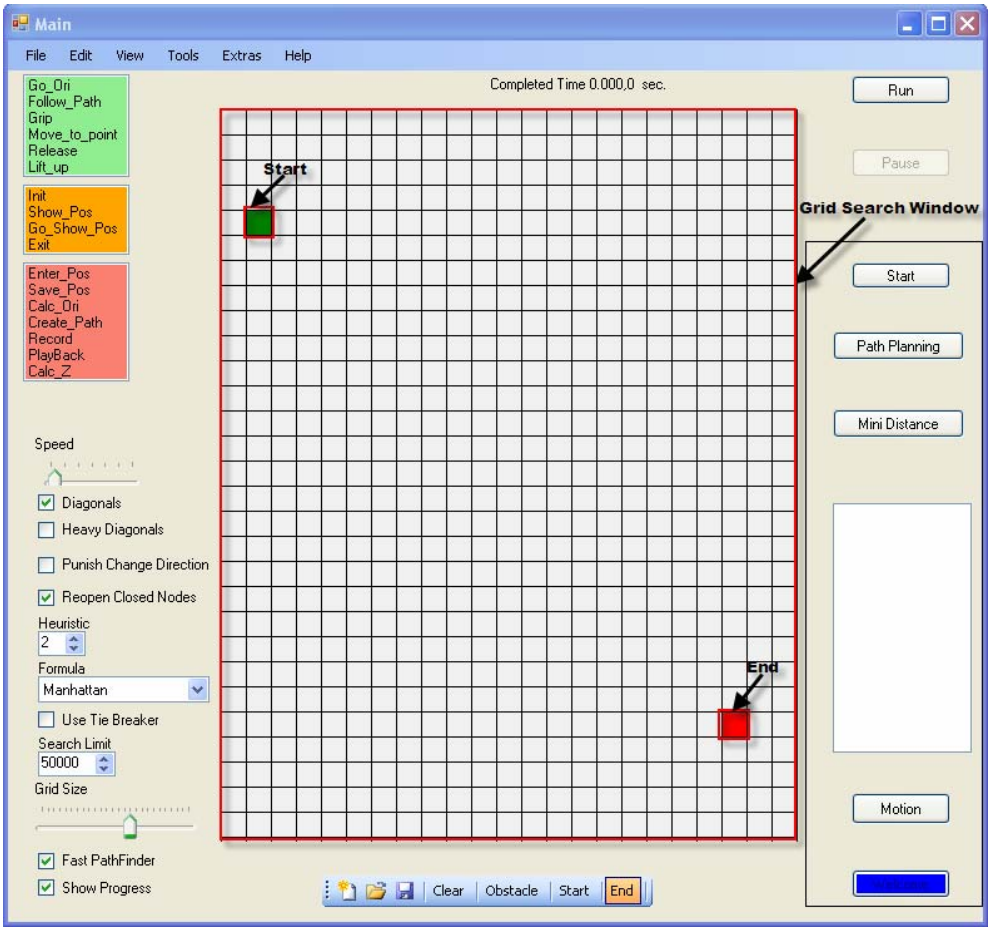


Figure 4-1 – Location of the grid search window for demonstration of A* algorithm

The first thing you should notice is that we have divided our search area into a square grid (see Figure 4-1). Simplifying the search area, as we have done here, is the

first step in path finding for A* algorithm. This particular method reduces our search area to a simple two dimensional array. Each item in the array represents one of the squares on the grid, and its status is recorded as walkable or unwalkable. The path is found by figuring out which squares we should take to get from A (start) to B (goal). Once the path is found, the manipulator should move from the center of one square to the center of the next until the target is reached. These center points are called “nodes”.

Figure 4-2, shows the “virtual space” form which pops-up when either Euclidean distance or artificial potential fields method is activated.

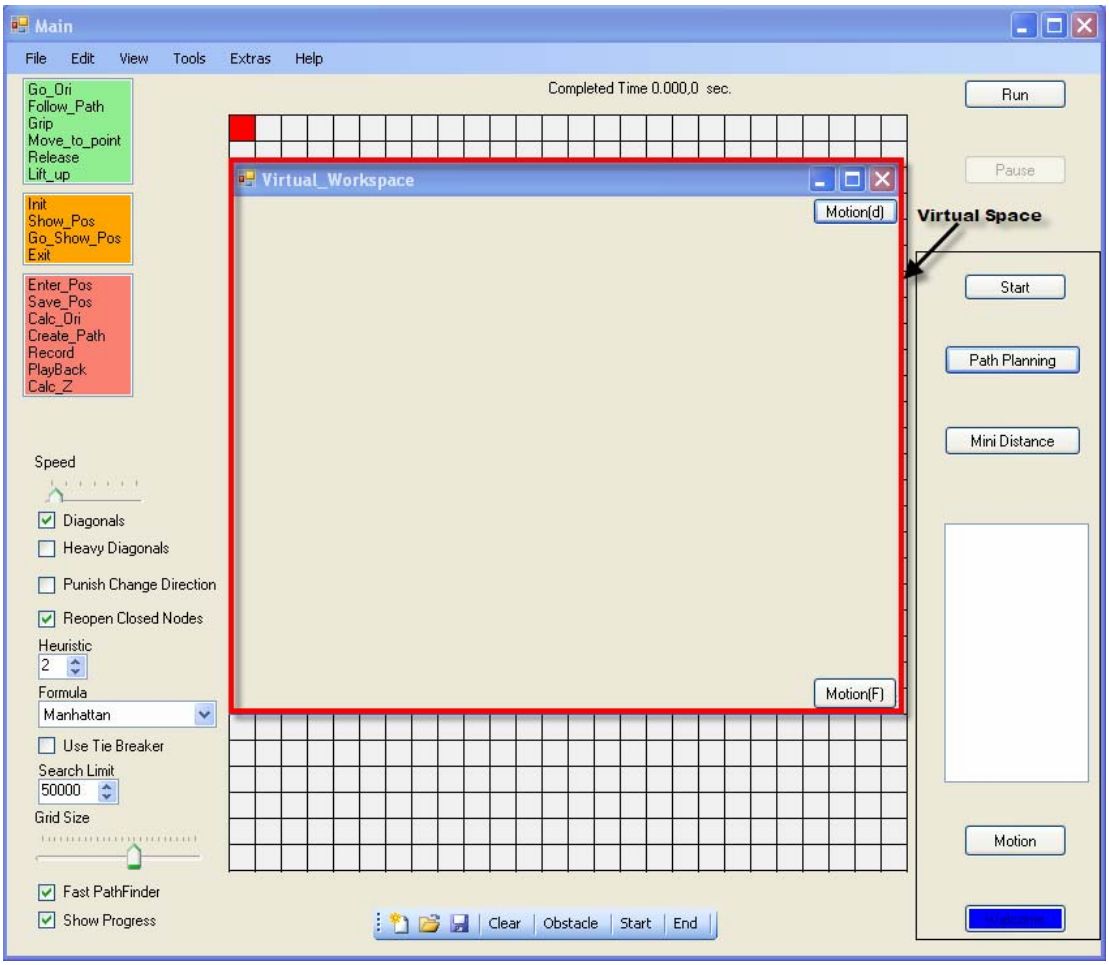


Figure 4-2 – Virtual space form for demonstration of EDA and potential field algorithm

4.2 Euclidean Distance Algorithm

Our first motion planning algorithm is the Euclidean distance algorithm, in which the particle with the nearest proximity to the destination is the one which is pushed first by the manipulator. This algorithm defines the geometric representations of all particles in the workspace, and that of the manipulator as well. The method allows planning algorithms to determine whether particle being moved by the manipulator is in collision with other particles or with obstacles. The idea is that, the particles that are closest to the destination are being pushed towards their proposed destinations.

Given an initial coordinate frame $F := (OXY)$ on our workspace with origin O and axes *X,Y*, the position of manipulator is determined by its coordinates (x_i,y_i). Then, if the position of the manipulator at any time (t) is ($x_i(t)$, $y_i(t)$) and let q(t) be the configuration variables, then

$$\begin{aligned} x_i(t) &= x(t) + d_i \cos(\theta(t) + \alpha_i), \\ y_i(t) &= y(t) + d_i \sin(\theta(t) + \alpha_i). \end{aligned} \tag{4-1}$$

where, d_i:=is the distance between original position of the particles and their destinations and α_i is the angle formed by the segments d_i and the positive horizontal axis. Let us introduce the distance function:

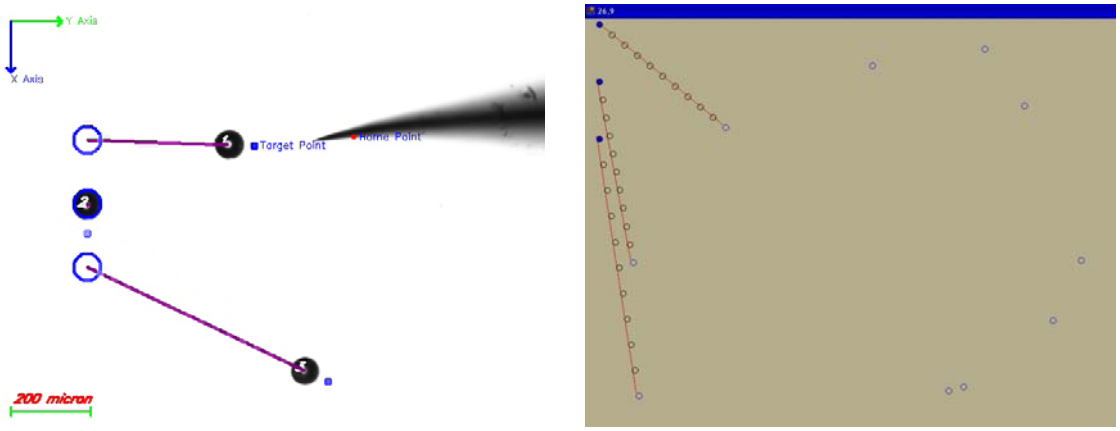
$$L(q(t), \dot{q}(t)) = \left[(\dot{x}(t) - d_i \sin(\theta(t) + \alpha_i)\dot{\theta}(t))^2 + (\dot{y}(t) + d_i \cos(\theta(t) + \alpha_i)\dot{\theta}(t))^2)\right]^{1/2} \tag{4-2}$$

Since for 2D manipulations the cost function is the distance between two points, our goal would be to minimize the function in equation (4-2), so that the manipulation is performed for the particles which are closest to the destination.

$$Path = \arg\min L(q(t), \dot{q}(t)) \tag{4-3}$$

In virtual space we consider the random distributed particles with the same physical properties. Our objective is to move the particles to their defined destinations without any collisions between them during the manipulation operation. Since the particles are spherical, in 2D the particles have the same radii; therefore there is no question whether the manipulated particle will fit well into their assembled locations. Assuming the above situation is true the manipulations begins with calculating the closest particle to the destination point in consideration. Then, a line connecting the defined destination location and the closest particle is drawn. Afterwards, the line is distributed into

segments depending on the structure of movement in the manipulation. Finally, the manipulation process starts by slowly pushing the particle towards the destination following the straight line. The Figure 4-3(b) below demonstrates how the process is implemented in virtual space with particles of radius 10 units. Similarly Figure 4-3(a) shows how the same algorithm can be applied in the MAW setup using semi-automated manipulation features already available in the setup.



(a) MAW experiment example (b) Virtual space implementation

Figure 4-3 – Comparison of scenario for motion of particle between MAW and virtual space of the window application

This type of manipulation considers the particles to be moved as if they have same physical properties (mass, area, and volume), therefore the necessary force required to move the particles is the same for all the particles in the workspace. However, in real situation the dimensions of the particle located on the workspace are of different magnitudes which prompts for the selection of which particle to move and when to move. In consideration of the above mentioned scenario, the necessity of the system to have the motion planning algorithm which has intelligence to allow manipulation of particle from their original locations to their defined destinations seems to be inevitable. Figure 4-4 demonstrates another application example of the Euclidean distance algorithm in the virtual space. The advantage of using this simple algorithm is that the obstacles (particles lying along the path of a possible candidate for manipulation) are given higher priority of manipulation before manipulation of the candidate.

Figure 4-4 – Motion of particle with the proximity based algorithm operating

4.3 Artificial Potential Field Algorithm

After the description of the Euclidean distance algorithm for path planning of particles with the same physical properties, it is also fair mentioning the scenario in which motion planning algorithm operates on particles with different physical properties. Then, the need to use artificial intelligence to allow manipulation of particles from their original locations to their defined destinations seems to be inevitable. In Artificial Potential Field method, an obstacle applies repulsive forces on the manipulator, simultenously the goal applies an attractive force to attract the manipulator and the particle being pushed towards its direction. Eventually the manipulator is forced to take the direction of the resultant force field.

One of the challenges with artificial potential field is the problem of local minima [8]; we had to find a way to get out of local minima in our solution or have no local minima at all. To accomplish this we had to modify the way in which we build our potential field. If the obstacle force experienced by the moving particle is as illustrated below.

$$\vec{F}_{obs} = -O \cdot \sum_{i=1}^{n} \frac{1}{d_i^2} \cdot \hat{r}_i \qquad (4\text{-}4)$$

where O is a constant scaling factor, n is the number of obstacles, d_i is the distance between obstacle i and the manipulator, $\hat{r}_i$ is the direction vector from moving particle to representative point of obstacle. Note that, in equation (4-4) O is divided by d_i^2, therefore, the obstacle force increases as the moving particle gets closer to the obstacle (as d decreases). The obstacle force comes into consideration once the obstacle reaches the perimeter of the dotted circle of Figure 4-5, otherwise the obstacle force is negligible. And the attaraction force between the moving piece and the goal is:

$$\vec{F}_{goal} = G \cdot d^2 \cdot \hat{r} \tag{4-5}$$

where G is similarly a scaling factor, d is the distance from moving particle to the goal, and $\hat{r}$ is the direction vector from manipulator and moving particle to the goal.

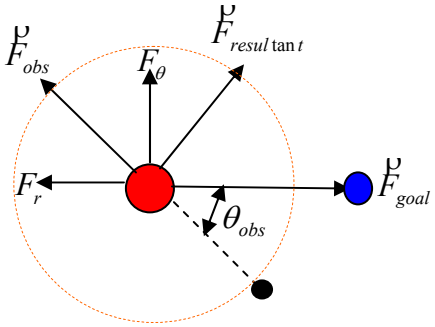


Figure 4-5 – Motion of particle with the artificial potential field algorithm

Using the outputs of the distance sensors, the net repulsive force is calculated and decomposed to its components, one along the direction of motion of the manipulator and one perpendicular to it (see Figure 4-5). If θ_{obs} is the angle between the direction of motion of the manipulator and the obstacle force, then

$$\begin{aligned} F_r &= \left|\vec{F}_{obs}\right| \cdot \cos(\theta_{obs}) \\ F_\theta &= \left|\vec{F}_{obs}\right| \cdot \sin(\theta_{obs}) \end{aligned} \tag{4-6}$$

For safe motion of the moving piece being pushed, the manipulator should try to keep the force component along its direction of motion, F_r minimum or ideally zero. This can be achieved by changing the orientation of the manipulator, since the force components are dependent on the orientation. To this end, a controller can be used for

the optimization. The rate of change of the force components with respect to the obstacle angle is,

$$\dot{F}_r = -\left|\vec{F}_{obs}\right| \cdot \dot{\theta}_{obs} \cdot \sin(\theta_{obs})$$
$$\dot{F}_\theta = \left|\vec{F}_{obs}\right| \cdot \dot{\theta}_{obs} \cdot \cos(\theta_{obs}) \qquad (4\text{-}7)$$

Then, the controls to drive the PZT 3-axis manipulator which hold the manipulator are selected as $u_r = \dot{F}_r$ and $u_\theta = \dot{F}_\theta$.

The techniques of using potential field can be seen as an interesting alternative for the A* algorithm that seems to be more popular in the current state of the art. The potential field method requires some serious calculations which are computationally expensive [9]; however, with modification in which computations for the potential field are done only in the vicinity of the moving particle, the resulting operation becomes less computationally expensive.

For the modification of the artificial potential field algorithm; first after selecting the particle to be manipulated, a line connecting the defined destination position and the particle to be manipulated is drawn. Then the particles close to the line joining the start and the end position of the manipulation are treated as obstacles as shown in the Figure 4-6 below. For software implementation in virtual space the moving particle only takes into consideration its distances from the particles along its path and its desired destination.

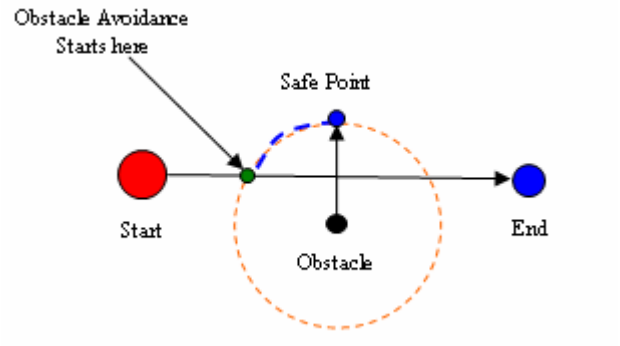


Figure 4-6 – Obstacle Avoidance path of the artificial potential field algorithm

Operation of the field oriented motion planning algorithm used for demonstration on our virtual space operates as follows:

Input: Destination (given as input pattern), particle to be moved (given a click input event on the virtual space of the window application) and obstacle coordinates (given as random particles dispersed on the virtual space).

Output: A path from origin to destination obtained through online pushing.

(i) Initialize the window application

(ii) Click to start the field oriented path planning

(iii) Click the particle to be moved

(iv) The algorithm will return, either the particle was successfully pushed towards its destination, or due to some discrepancies the manipulation of the particle failed.

Figure 4-7 demonstrates the full mode of operation for the modified version of the potential field algorithm in the virtual space of our window application.

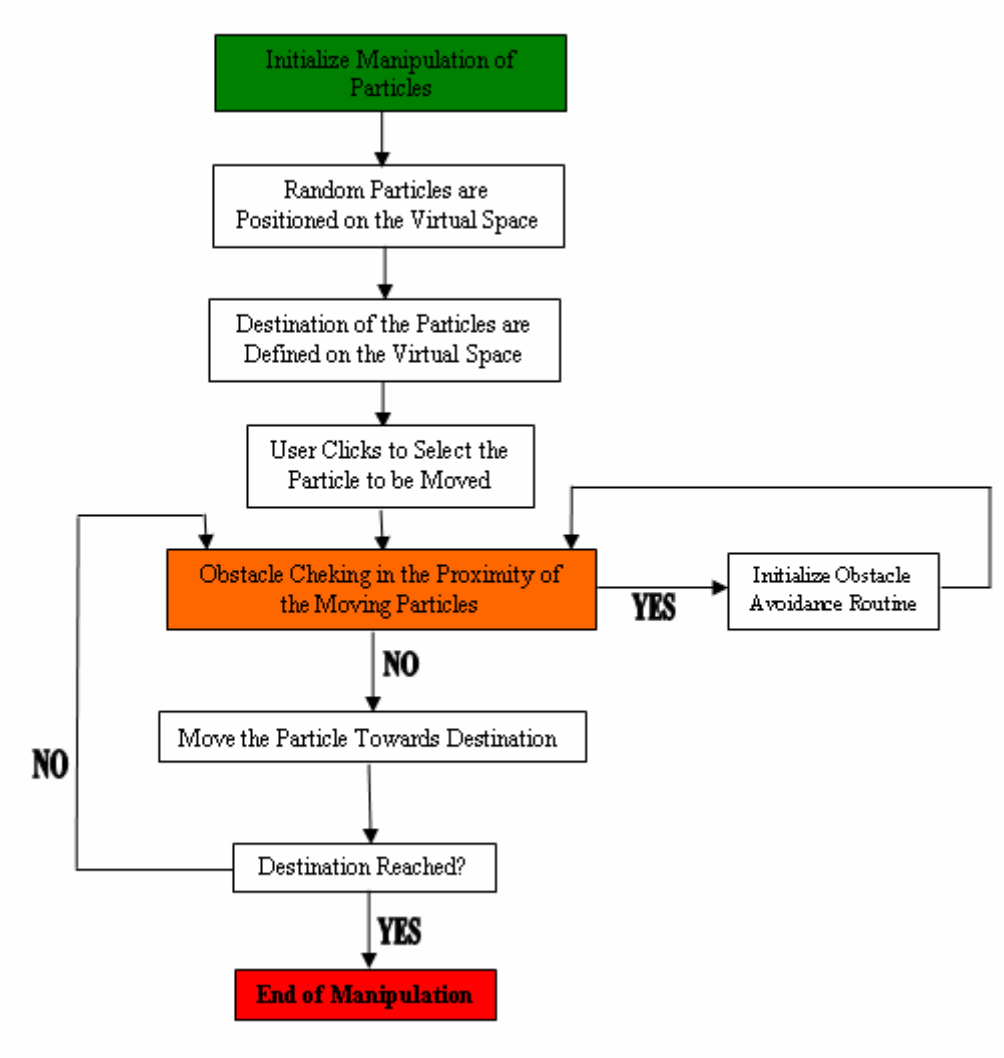


Figure 4-7 – Structure of artificial potential field algorithm with obstacle avoidance

4.4 Graph Traversing A* Algorithm

Finally, we implement the A* algorithm; a popular path finding algorithm used to find a shortest path. A* algorithm incrementally builds all routes leading from the starting point until it finds one that reaches the goal. But, like all informed search algorithms, it only builds routes that appear to lead towards the goal. In order to determine which routes will likely lead to the goal, A* algorithm employs a heuristic estimation of the distance from a given point to a specified goal. We can find an unobstructed path through configuration space by discretizing the configuration space and then employing some form of search algorithm. Considering all possible actions and all possible states to which the manipulator can transition is un-realistic because the size of the search space would be too large or infinite. Instead we chose to represent the workspace as a set of discrete states and we select a subset of manipulator's actions that correctly transition the manipulator between those states while not limiting the manipulator's capabilities too severely.

Methods for discretizing the configuration space take on two general forms: skeletonization and cell decomposition. Skeletonization reduces the configuration space to a one-dimensional space consisting of a network of connected curve segments through free space, collectively called a skeleton. Cell decompositions break the configuration space into adjacent regions (spaces). For our system we chose uniform grid cell decomposition at multiple discrete orientations for its flexibility and simplicity.

We create discrete states in the configuration space by laying a two-dimensional grid over the workspace and then considering only a discrete set of orientations. The discrete coordinates and orientations used for path planning need not correspond directly to the discretization used to calculate the configuration space. For example, it might be advantageous to search through a very simplified state space but then check whether an action is obstructed at a much higher resolution.

Search is the process of exploring sequences of actions to determine a sequence that leads to a desired goal state. An action performs a transition from one state to another. A sequence of actions from the initial state is called a search node. The search nodes form a search tree, where the fringe of search is at the leaves. In motion-planning the goal state is some desired point in configuration space and the actions represent motions the manipulated particle is due to perform. We approximate initial state and the

goal state by converting them to the nearest state in the discretization of the configurations space being used, and then search for a sequence of unobstructed actions that, when completed, end at the goal state.

The search algorithm is initialized by placing the goal state in a priority queue. The algorithm then proceeds by removing the first node on the priority queue and checking if the goal has been reached. If it has not, the node expanded into its neighboring nodes, which in turn placed on the queue. The search continues until the goal state is found, the search space is exhausted, or some other stopping criterion occurs, such as a timeout.

The searching procedure is conducted as illustrated in Figure 4-8. In this figure, the dark green square in the center is your starting square. It is outlined in light blue to indicate that the square has been added to the closed list. All of the adjacent squares are now on the open list of squares to be checked, and they are outlined in light green. Each has a gray pointer that points back to its parent, which is the starting square.

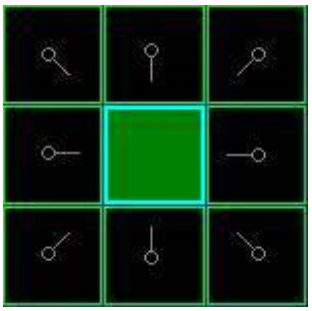


Figure 4-8 – Demostration of how a search is conducted from a node [18]

Depending on the position of the particle, the position of the goal, and the position of the obstacles in the workspace, there may not be a legal path from the particle's initial state to the goal state. This condition can be detected by checking whether or not the search queue is empty when the search terminates and it should be reported to the procedure's caller.

Different priority queue implementations yield different search behaviors. A priority queue that orders search nodes based on the cost:

$$f_{st}(n) = g(s,n) + h(n,t) \tag{4-8}$$

where $g(s,n)$ is the cost of the path from the start node s to a node n, and $h(n,t)$ is the estimated cost from a node n to the goal node t yields the efficient heuristic search

algorithm called A* [13]. A* returns an optimal path sequence if the heuristic used to estimate the cost to the goal is admissible,

$$h(n,t) \le g^*(n,t) \tag{4-9}$$

or in other words, if the heuristic function h from node n to the goal node t is always an underestimate of the optimal cost $g^*(n,t)$ from n to t (the "*" stands for optimal or shortest). A commonly used admissible heuristic in motion planning is the straight line distance to the goal, although other heuristics are possible (see Figure 4-9).

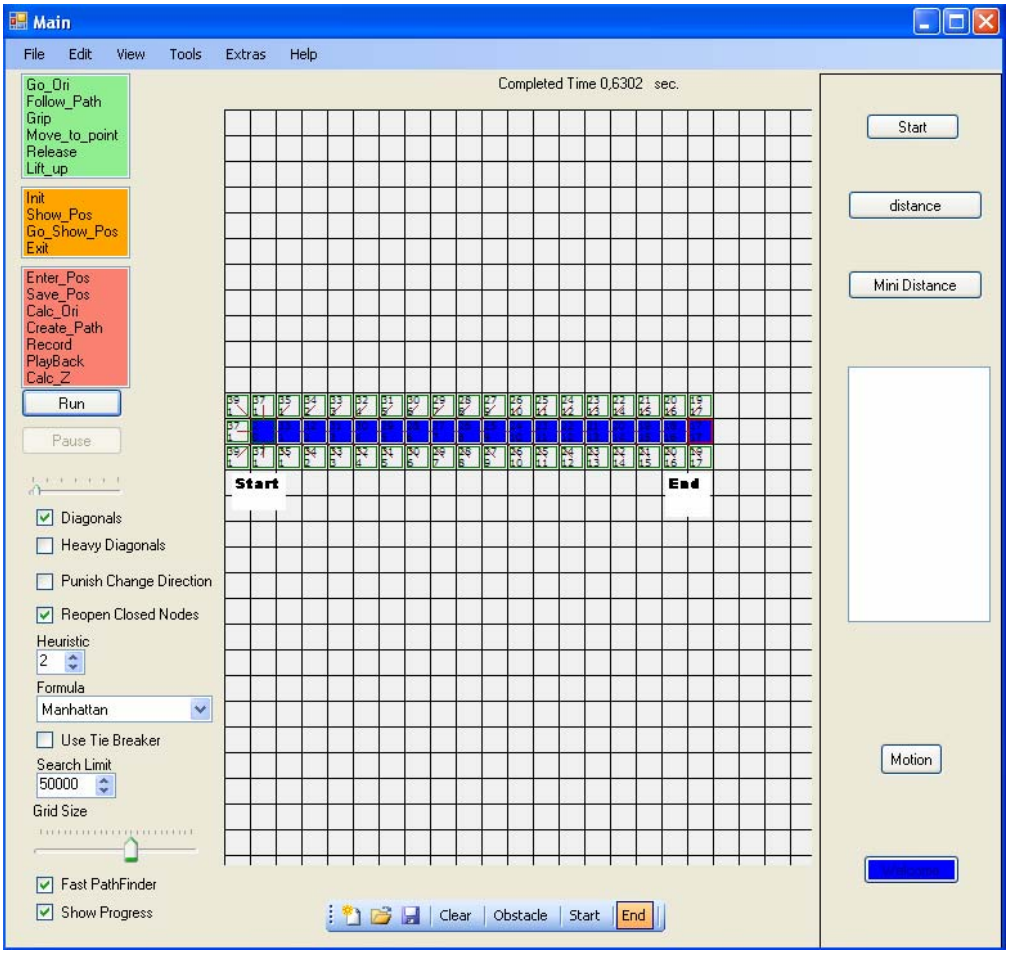


Figure 4-9 – Demostration of straight line travel from start (origin) to end (destination)

Search is computationally expensive, as it generally exponential in the depth of the search in both space and time. A perfect heuristic would change the running time to be linear, but perfect heuristics are difficult to come by. We can speed up the A* algorithm and reduce the size of the search problem to the size of the state space by eliminating loops from the search tree. In other words, we want to avoid considering the same actions from a given state multiple times and also stop pursuing multiple action

sequences that lead to the same state. The nodes that have already been expanded are sometimes called the closed list or the expanded list. The nodes still on the queue are called the open list, visited list, or the fringe.

Changing the search algorithm to only expand nodes that have not been expanded before changes the optimality condition on the search heuristic. The heuristic must also be consistent, which means it must obey that triangle inequality.

$$h(m,t) \leq g(m,n) + h(n,t) \tag{4-10}$$

for an intermediary node n. Fortunately, admissible heuristics that are not also consistent appear to be somewhat rare, and the straight line distance heuristic is both admissible and consistent.

Keeping the entire search tree in memory is expensive but feasible for reasonably sized local path planning problems. We can reduce the space cost by only keeping in memory one node per state and then also checking whether a node to be put on the queue represents a state that has been put on the queue before. If node representing the same state has already been put on the queue is less than the cost of the new node, then the new node need not be put on the queue as well. Otherwise the cost of the node in memory should be updated with the smaller cost, and the new node should be put on the queue and the old node should be taken off. Removing the old node may not be practical depending on the priority queue implementation, but leaving it on the queue does not affect the order of node expansion so we do not bother with this step in our implementation.

Figure 4-10, shows the implementation of the A* search algorithm with the obstacles located along the way.

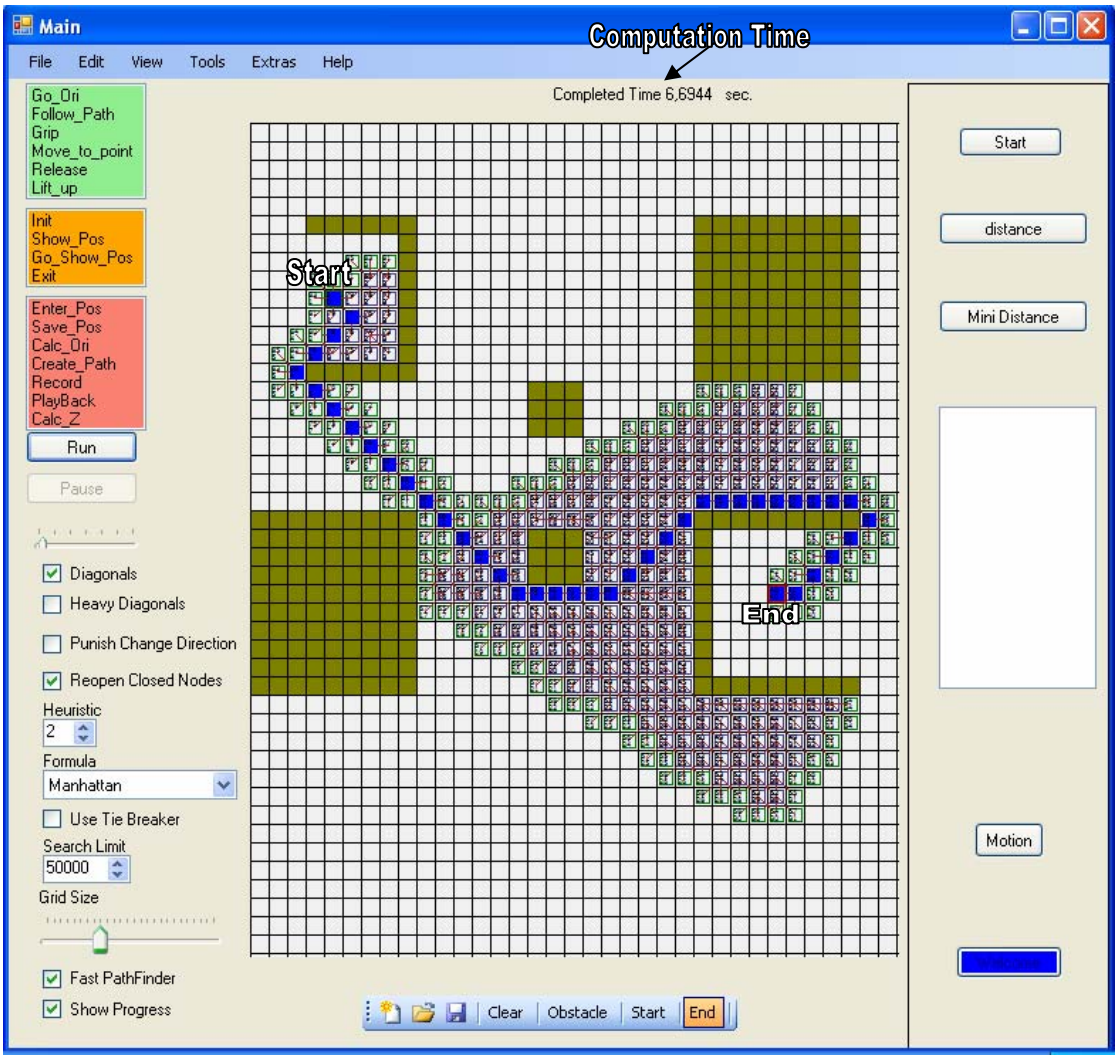


Figure 4-10 – Demostration of obstacle avoidance in A* path planning; grid in blue is the optimal path

In A* algorithm, heuristic function; is a constant that will affect the estimated distance from the current position to the goal destination. A heuristic function is used to create an estimate of how long it will take to reach the goal state. The best estimate is generally the one which gives the shorter path with smallest terrain cost.

Formula is the equation used to calculate the heuristic. Different formulas will give different results: some will be faster, others slower and the end may vary. The formula to be used depends strongly on the A* algorithm's use.

When A* is finding the path, sometimes a “tie breaker” is used since the algorithm may find many possible choices for the same cost and destination. The tie breaker setting tells the algorithm that when it has multiple choices to research, instead it should keep going. As it goes, the changing costs can be used in a second formula to

determine the “best guess” to follow. Usually, this formula is incrementing the heuristic from the current position to the goal, multiplied by a constant factor.

Obviously, this requires each of the paths to be searched to the very end to determine which path possesses the deepest node; and this takes more time. Consequently, A* is frequently implemented with various ‘tie-breaking’ heuristics. A different way to break ties is to prefer paths that are along the straight line from the starting point to the goal: This urges for computation of the vector cross-product between the start to goal vector and the current point to goal vector. When these vectors don't line up, the cross product will be larger. The result of such computation will give some slight preference to a path that lies along the straight line path from the start to the goal when there are no obstacles.

If $A = \begin{bmatrix} dx_1 & dy_1 \\ dx_2 & dy_2 \end{bmatrix}$ is matrix for node changes,

where dx_1 = currentX = parentNode.X - end.X;

dy_1 = currentY = parentNode.Y - end.Y;

dx_2 = goalX = start.X - end.X;

dy_2 = goalY = start.Y - end.Y;

Then, in such cases the formula below is used.

$$Heuristic = Heuristic + abs(\det A) * 0.001 \tag{4-11}$$

Best choice of heuristic, lead to obtaining the best choice of a path in motion planning using A* algorithm. For further details on heuristics used in the motion planning using A* algorithm please refer to [18].

4.5 Conclusion and Discussion

In this chapter, issues related to the motion planning algorithms of the microassembly workstation were presented. However, in the following chapter we provide results when we studied the feasibility for constructing 2D microparticles/ microstructure autonomously using image processing and motion planning algorithms in the microassembly workstation.

5 EXPERIMENTS AND RESULTS

5.1 Introduction

In testing the reliability of the window application software for different microassembly tasks, several experiments are implemented in different modes. These experiments will be elaborated in the following sections. Firstly, commands for single microassembly operations are realized by giving single command, and then allowing the system to execute the command before a next microassembly command is given on the window application. Then, motion planning algorithms are implemented to demonstrate the ability to reduce human intervention by allowing several commands to take place sequentially to achieve certain tasks. The operator chooses the motion planning algorithm to be implemented then using the algorithm the path used to intelligently manipulate the particle towards its destination is defined. Only in the artificial potential field algorithm the operator has also to specify the particle to be manipulated and in the A* algorithm operator has to give few specifications before the motion planning begins.

Experiments related to these two approaches and results will be shown in the following sections. Evaluations will be made according to the results achieved by the experiments and the chapter will be concluded with some discussion about the results of the experiments.

For the realization of micromanipulation and microassembly tasks, it is necessary to visualize and sense the environment. Position and orientation of microparticles with respect to each other and the manipulation tool can be defined by using a vision system. Since vision provides fast and contact less information extraction, it is suitable for visualizing the microworld and providing position and orientation data for the micromanipulation and microassembly tasks.

For the automated microassembly tasks, position and orientation information of specific manipulation tools or micro parts must be extracted by means of vision system.

The information named as visual features, is used in the control loop of the manipulation system. Thus, vision based control is attained.

However in the window application's virtual space the mode of operation is a bit different. Since our goal is to provide the motion planning for the realization of micromanipulation and microassembly tasks, the software does random calculations for the x and y coordinates of the particles each time the motion planning algorithm is initialized. The application operates under the assumption that all the information about the particles is correctly obtained from vision system. Therefore, the experiments given in this thesis work are graphical simulations showing the path in which the micromanipulators should follow in orienting and pushing the particle towards its destination.

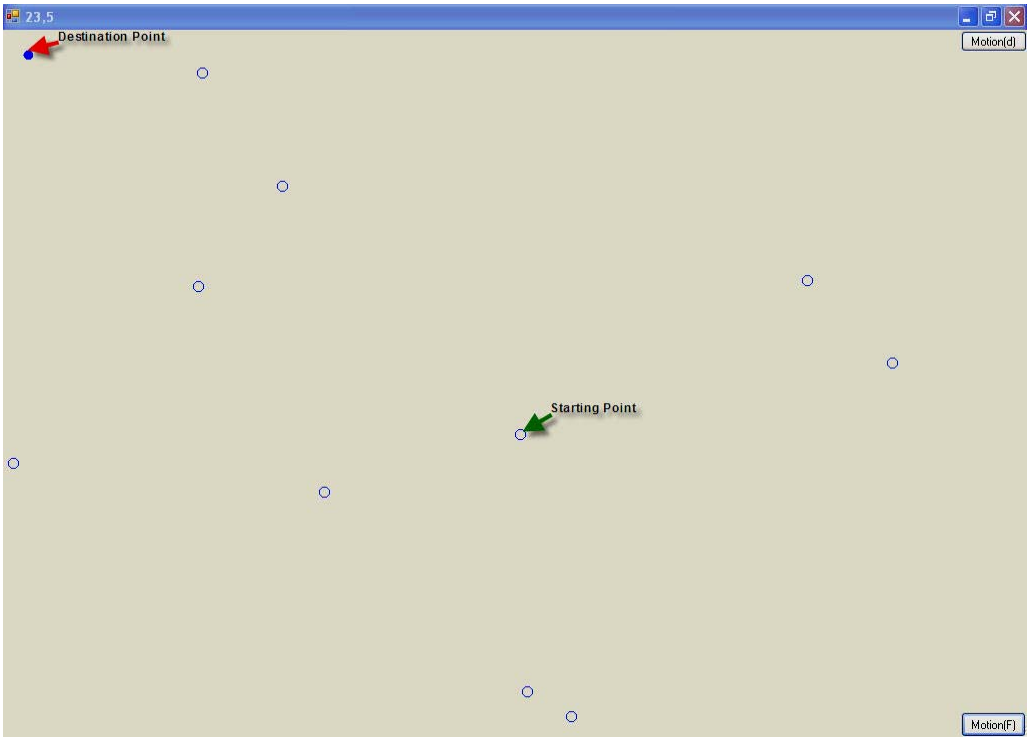


Figure 5-1 – Random Position of the particles in Virtual Space

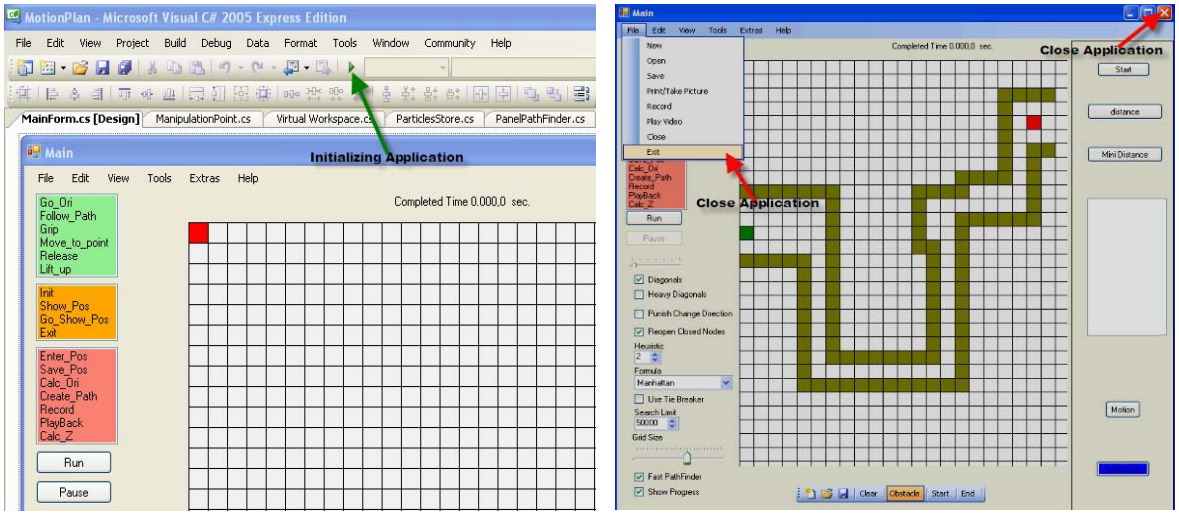
Figure 5-1 shows the random distribution of the particles on the virtual space prior the selection of planning algorithm (EDA or Potential Field Algorithm). For the simulation of motion planning algorithm using A* the "grid search window" (see Figure 4-1) in the middle of the main form of our widow application is used for demonstrations of the most optimal path. In the following sections we will cover the experimental details for the single mode operations and motion planning algorithm while running the window application software.

5.2 Commands for Single Operations of Microassembly

5.2.1 Definition

In the single operations for the microassembly, single command are given and then allowing the system to execute the command before a next microassembly command is given on the window application.

For initializing the application operator has to click the “play” button in green. This initializes the main form for the motion planning algorithms. Contrary to that, in closing the application operator has to click “File” and select “Exit” in the pull-down menu, or simply click the “close” button on the upper right corner of the main form window (see Figure 5-2(a)-(b) for elaborations).



(a) Initializing Manipulation Path (b) File-Exit or “close” terminates the application

Figure 5-2 – Experiment showing single mode commands

The “File” menu of the pull-down menu contains several commands such as “Save Video” which saves the video of an activity taking place on the active window on the screen. While the “Play Video” retrieves the video saved in the specified memory location and plays the “avi.file” on the screen using “Windows Media” form. In addition, “Print/Take Picture” menu captures snapshots of the activities taking place in the main form window. Lastly, the “Exit” command closes the main form and exits the application.

The “View” menu contains commands such as “Zoom”, “Resize” and “Full Screen” which helps to resize and reshape structure of the main form window of the window application. Also, “Tools” menu of the pull-down menu comprises of commands such as: “Create Path” command for initializing the A* graph traversing algorithm is found, along with “Slow/Fast” tool to change the speed of the path finding algorithm, and “Resume” tool to continue the path finding algorithm when in the “Pause” mode.

The “Extras” menu contains the “Particle Select” which shows the x and y coordinates of all the particles located on the virtual workspace of the main form. While “Particle Show” gives the x and y coordinates of the specific particle selected on the virtual space; usually it also demonstrates the coordinates of the particle selected for motion planning using potential field algorithm. In the “Extras” menu a shortcut to initialize the artificial potential field algorithm is also located. Other commands such as demos on how to run the application are yet to be implemented in the software in order to increase human computer interaction functions.

5.2.2 Implementation and Results

EDA and PFA algorithms operating on the virtual space are initialized by drawing random particles on the workspace. Therefore, when the user selects “Particle Select” (Figure 5-4) from “Extra” menu tools of the main window a list of x and y coordinates of all the particles located on the virtual space is given on the list box of the main form of window application as shown in Figure 5-3 below. The “Action Control Commands” contains the *Path Planning*; command used to initialize the virtual workspace form, so that EDA and potential field oriented motion planning algorithm can be used to determine path for manipulation of randomly distributed particles.

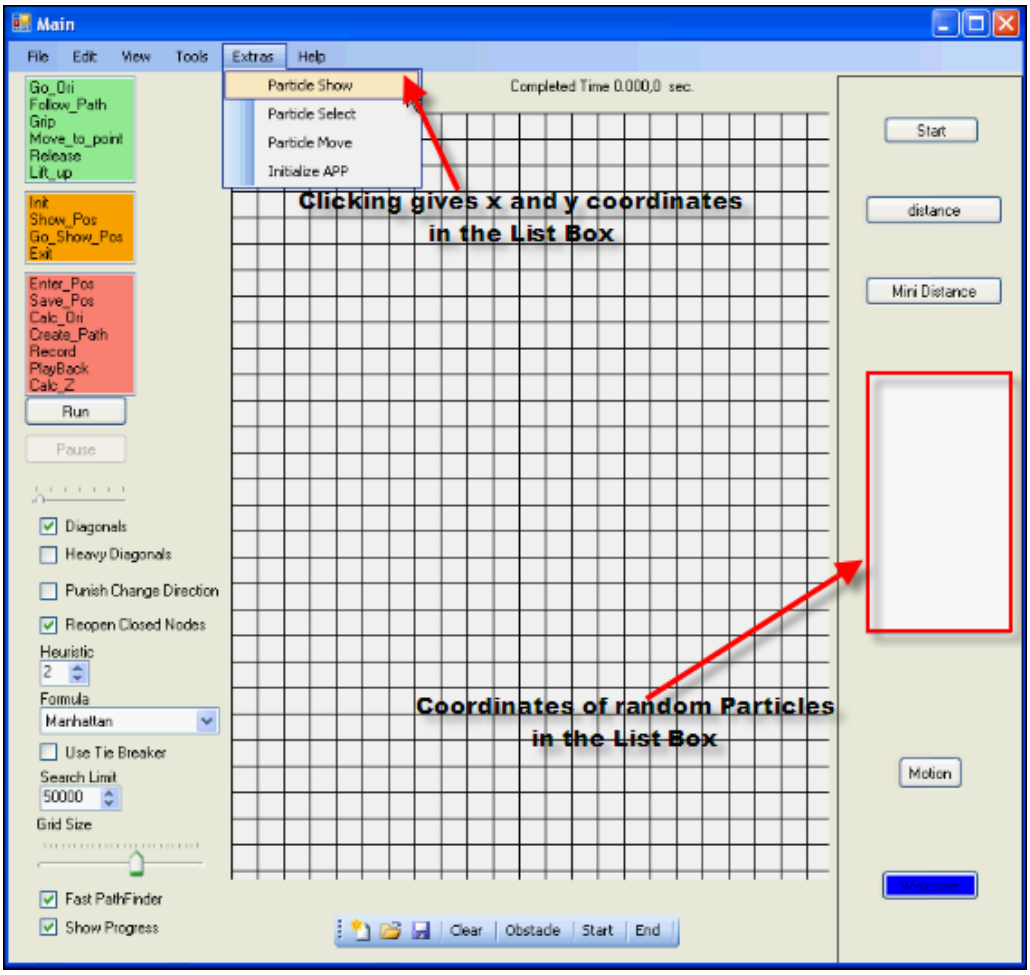


Figure 5-3 – Window showing "Particle Show" which gives the x and y coordinates of the clicked particle and list box of the main form gives the list of all x and y coordinates of the random particles.

For the EDA each particle is intended to be moved to the closest target point in the workspace. However, in order to guide the operator, in pushing a particle which is obstructed by other particles on its trajectory, operator acts by simply clicking on particle on the screen, then user has to acknowledge by clicking on the message box which gives the x and y coordinates of the selected particle. Alternatively, when the "Particle Show" from "Extras" tool command is selected a message box pops-up to give x and y coordinates of the selected particle.

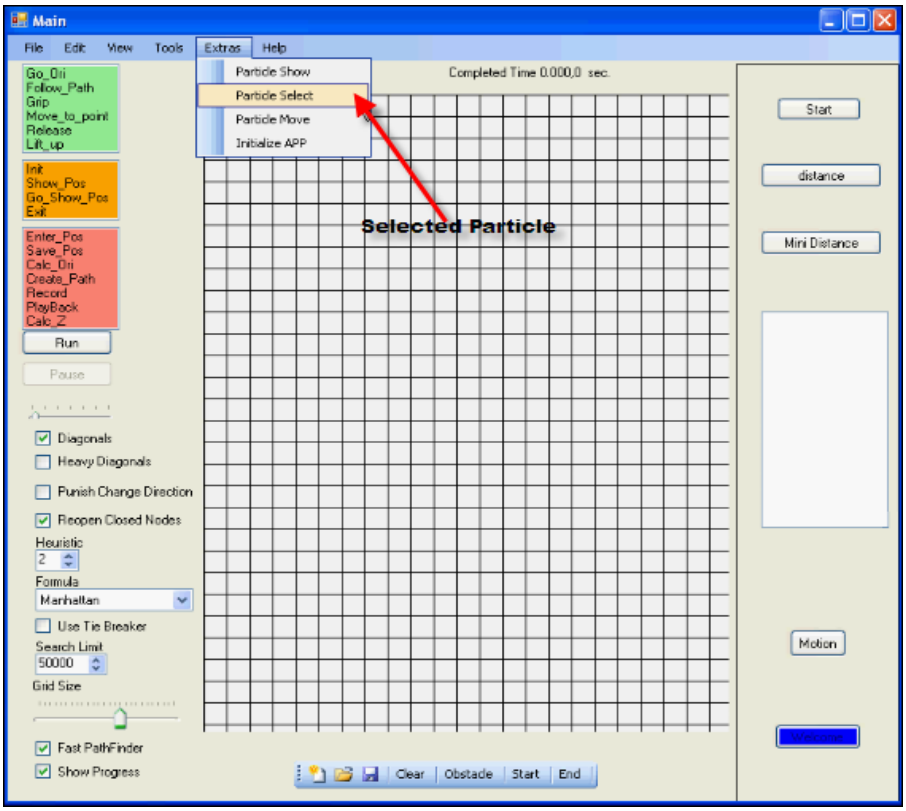


Figure 5-4 – "Particle Select" command shows x and y coordinates of all the random particle

5.3 Execution of Motion Planning Algorithms in Microassembly

5.3.1 Definition of the Task

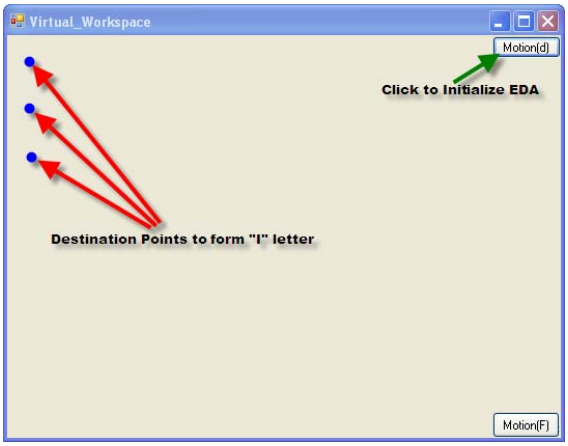
In semi-autonomous manipulation, the system initialization includes the selection of the particle and the desired destination point. Later, a line connecting the center point of the selected particle and the destination particle is drawn. Then, step motion value for pushing the particle and the selected point are calculated. Finally, the tip of the probe is moved in steps towards the destination point in order to achieve manipulation. The steps are repeated until the center of the selected particle coincides with the desired point.

The most feasible trajectory for a particle to its target position by pushing is simply a line denoting the closest path to the destination. In that context, the operator should choose the suitable part to be pushed and the destination point considering the issue that the semi-automated assembly procedure does not include motion planning so

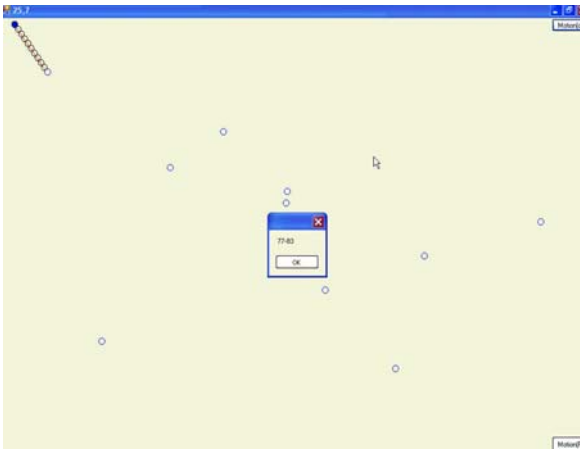
that there exists nothing as an obstacle between the particle and the target point. In our thesis work this feature is added as Euclidean distance planning algorithm in which the particle are manipulated depending on their proximity to the destination point.

5.3.2 Microassembly Examples

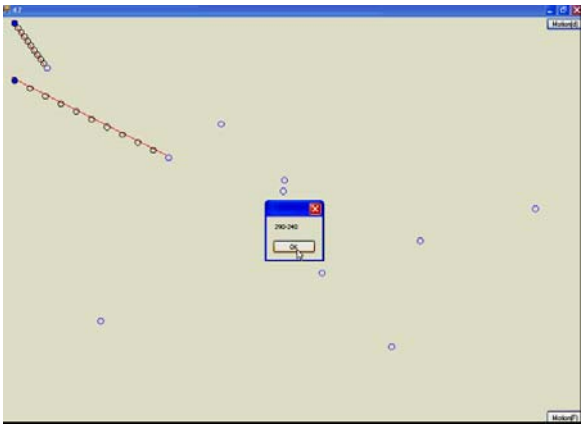
EDA Results:



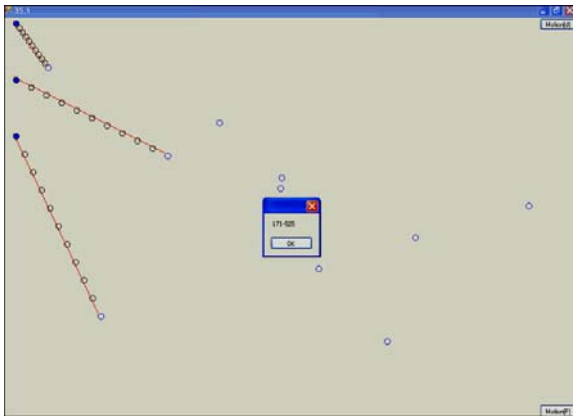
(a) Initializing Manipulation Path



(b) First Particle's Path



(c) Second Particle's Path



(d) ThirdSecond Particle's Path

Figure 5-5 – Snapshots for Path Planning using EDA

Figure 5-5 (a) shows the pattern about to be assembled by using EDA motion planning algorithm. Figure 5-5 (b)-(d) presents the consecutive motion planning path for assembling each particle to formulate an "I" like pattern on the left corner of the

virtual workspace. In addition, Figure 5-6, presents a complete path plan for different set of random particles on the virtual workspace.

More results:

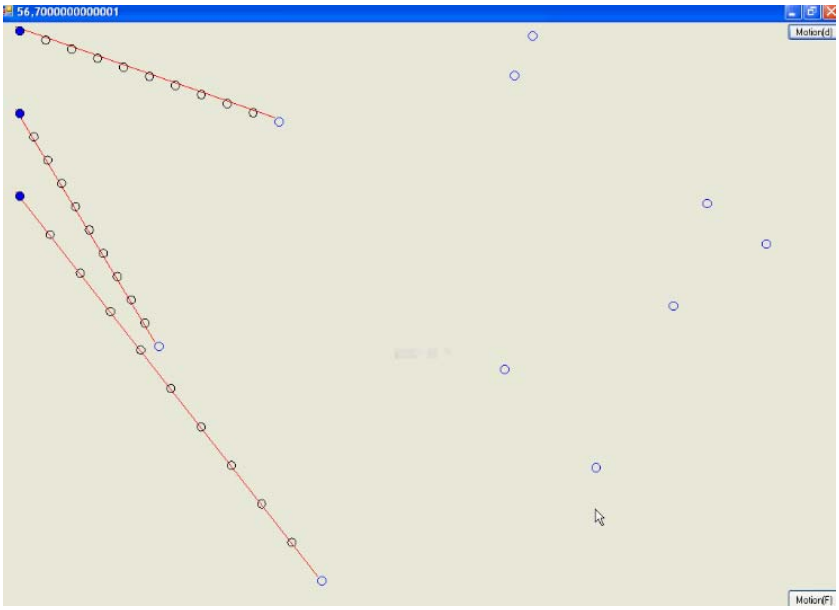


Figure 5-6 – Pattern Assembly using EDA

PFA Results:

In [16], several semi-automated tasks are generated and implemented with the microassembly workstation. One of the main problems for the realization of the experiments is to find suitable microparticles to be used in the microassembly operations; in our virtual experiments the problem is solved depending on the scenario (distribution of particles on the workspace). If the particles to be manipulated have different physical properties artificial potential field method is used to determine the path for manipulation of particles from their original locations to their defined destinations. To initialize the algorithm the operator has to select the particle to be manipulated out of crowd of randomly distributed particles on the virtual workspace. Then, manipulation is conducted as shown in Figure 5-7. In that experiment, operator selects the particle, while the destination point of the particle is provided (known). This selection is made by the operator for every particle's path planning. Then, after particle selection is done, user has to acknowledge the coordinates of the particle given on the message box on the screen and the rest of operations are executed automatically.

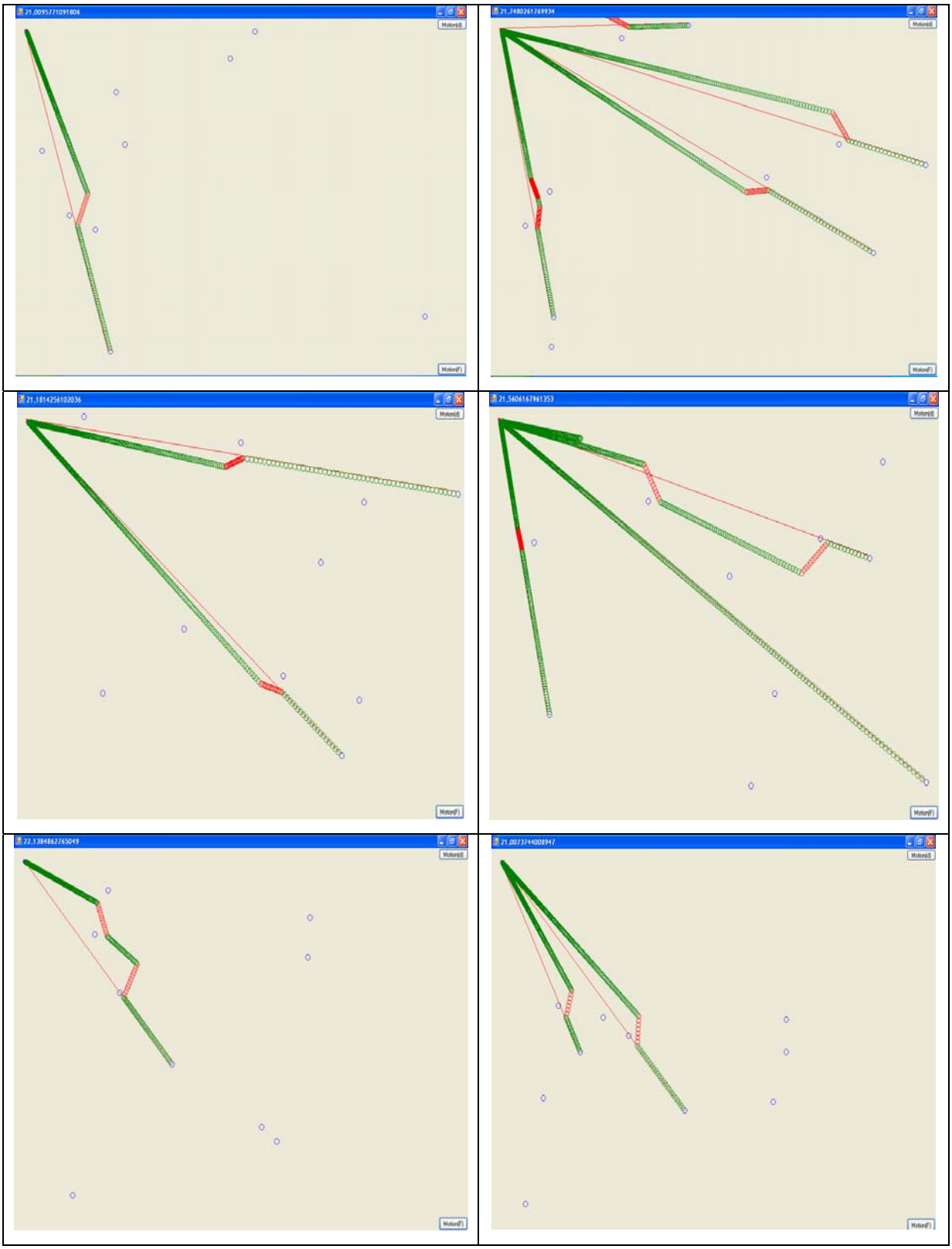


Figure 5-7 – Potential Field oriented motion planning algorithm

Figure 5-7 presents results for the object avoidance algorithm using potential field built around obstacle to guarantee that the particle to be manipulated doesn't collide with other particles on the workspace.

A Results:*

In A*, the main features necessary to perform motion planning for the manipulation tasks are pre-programmed and the operator simply has to define some of the parameters necessary for the assembly. Figure 5-8 shows results of A* motion planning algorithm for a path along the corridor.

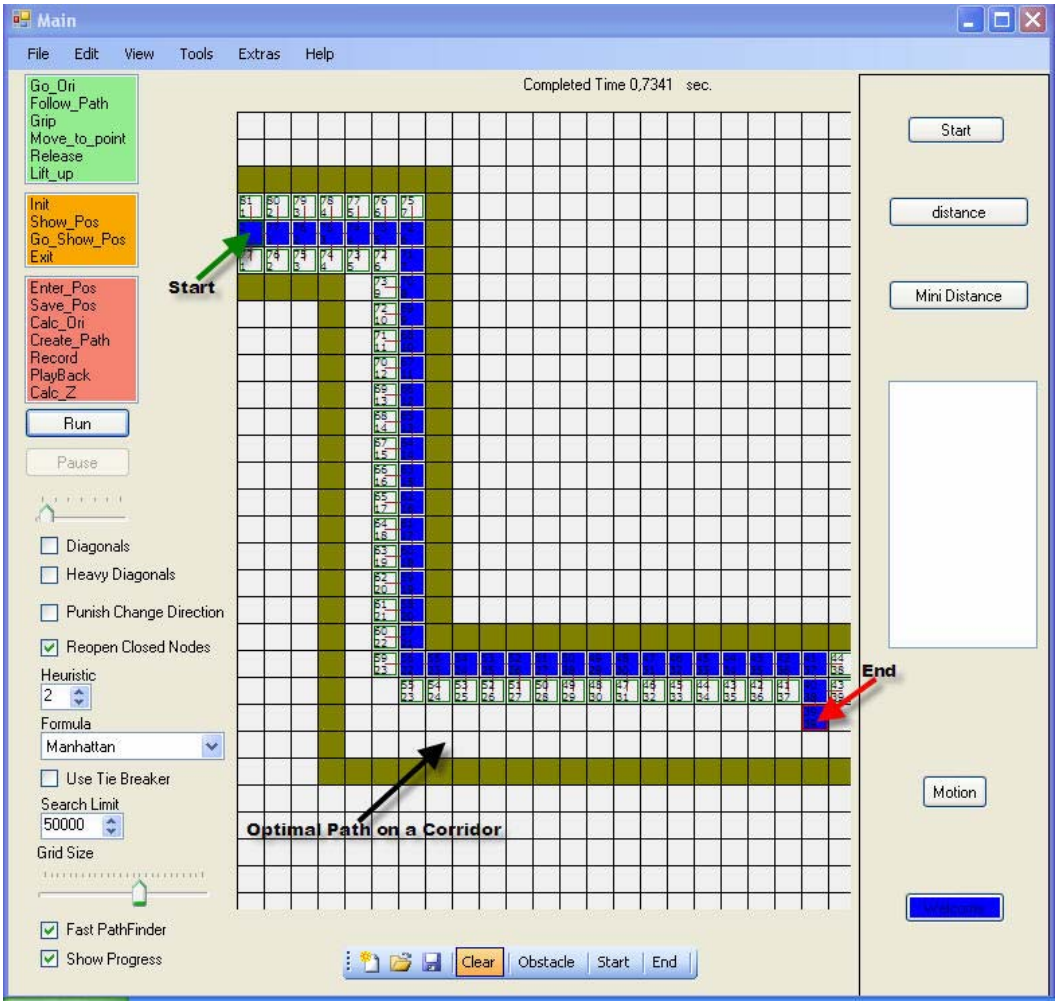


Figure 5-8 – A* algorithm motion planning to demonstrate the optimal path on the corridor

For more results of A* algorithm in different scenarios see (Figure 5-9). The results in Figure 5-9 only show the grid search window; A* algorithm is functional for wall avoidance, optimal path planning around obstacles, corridor following, and also it returns "no path" when path is not found during path planning process.

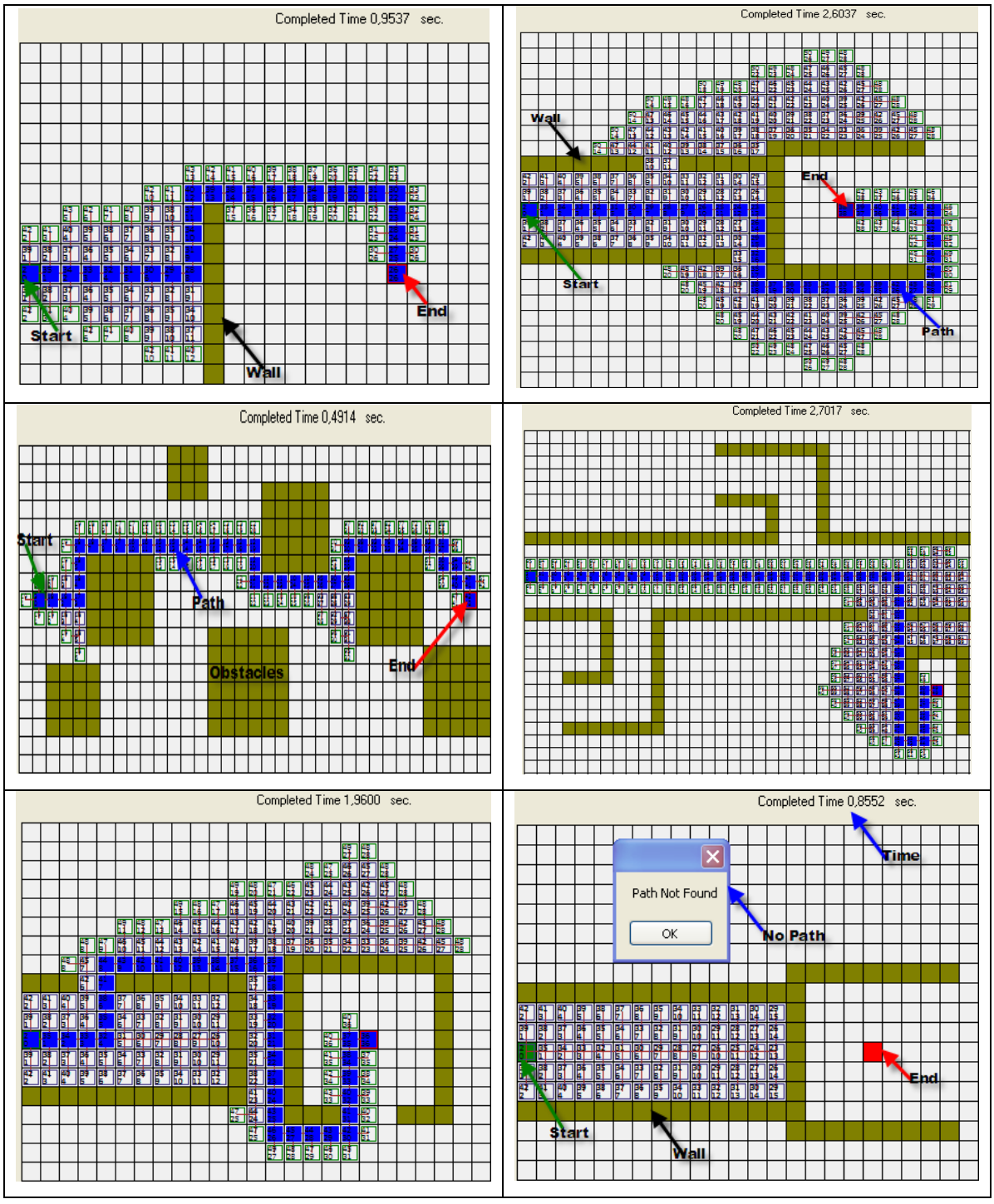


Figure 5-9 – More results of A* algorithm on different scenarios using different heuristics

5.4 Results and Discussion

The challenges of microassembly workstation regarding the manipulation operation like obtaining depth information, sticking problems, effects of forces and

many issues concerning sample preparation still hinder full operation of the motion algorithm on the microassembly workstation setup. These issues should be carefully dealt with for a successful automated assembly operation.

Other issues involved in the plan execution is the process of taking a motion plan from the motion planning model and then sending the correct control to the manipulators and stages so that they perform a desired task. The plan execution unit must execute the planned actions as accurately as possible and also deal with failure conditions. Issues that may arise during such process are such as latency; if the latency is too large the manipulators may drift from intended path or yield unexpected results, obstructed path; in which even if motion planning finds a valid path to the goal this may become obstructed due to previous unobserved obstacles becoming visible, new goal location; user may change the location of the goal when this happens the system must abandon any current execution plans and proceed towards the new goal location, and finally lost target; in which particle may have to move away from the goal to avoid obstacle(s) this may cause the particle to move outside the virtual space boundaries or cause the goal to go out of range [13].

6 CONCLUSION

In this work, first the introduction of an open-architecture and reconfigurable microassembly workstation for efficient and reliable assembly of micromachined parts was presented. Then, software description of the microassembly workstation was presented. Furthermore, in order to reduce human intervention in the precision manipulation, we developed and implemented several motion planning algorithms which could be rigorously applied in our setup. It should be noted that prior the introduction of the motion planning algorithms, the system lacked the ability to avoid obstacles, especially in the case semi-autonomous manipulation case.

The realized microassembly workstation represents an important step towards the automatic, autonomous assembly of micrometer-sized parts by means of sensory feedback. With the integration of motion planning algorithms, the system will have the ability to function in an automated mode where human intervention will be greatly reduced.

This thesis also presents how the window application was prepared using object oriented programming environment set of commands to achieve several tasks, along with motion planning algorithms for autonomous manipulation of objects in virtual space. Experiments regarding the implementation of conventional euclidean algorithm were demonstrated; in this case the solution for planning problem is trivial since the particle closest to the goal is the one which is pushed towards its destination.Then, the application of artificial potential field control algorithm was implemented with modifications in which the manipulator only reacts to obstacles in its proximity. Lastly, application of motion planning algorithm A* was demonstrated with feature which facilitate the object avoidance in manipulaton of objects from one point to another in the virtual space. Since the objective is to reduce human intervention in the precision autonomous manipulation, application of motion planning algorithms brings valuable ingredient to the Microassembly Workstation setup. All in all, this type of work

platform can be seen as a step in using a standard platform in the manipulation of micro-entities using micro-scale manipulation tools. Results of this thesis have been submitted as conference paper [20].

Future work will include implementing the features of the work platform obtained from this thesis to the Microassembly Workstation (MAW) setup. Also, the assembly process proposed in this thesis can be improved further by introducing autonomous algorithm which can choose most optimal algorithm. The autonomous algorithm for motion planning will further reduce human intervention observed in the semi-automated microassembly processes.

REFERENCES

[1] C. Belta, "Symbolic Approaches for Robot Motion Planning and Control," *In Proc. Of the IEEE Int. Conf. on Robotics and Automation*, Rome, Italy 2007

[2] J. H. Makaliwe and A. A. G. Requicha, "Automatic Planning of Nanoparticle Assembly Tasks," *Proc. of the IEEE Int. Symposium on Assembly and Task Planning*, pp. 288-293, Fukuoka, Japan, May 2001.

[3] C. Pawashe, and M. Sitti, "Two-Dimensional Vision Based Autonomous Microparticle Manipulation using Nanoprobe," *Journal of Micromechatronics*, September 2006.

[4] F.Markus Jönsson, "An optimal pathfinder for vehicles in real-world digital terrain maps," *Master's Thesis, Royal Institute of Science Stockholm*, Sweden, 1997.

[5] M. Kloetzer and C. Belta, "Managing non-determinism in symbolic robot motion planning and control," *in Proc. Of the IEEE Int. Conf. on Robotics and Automation*, Rome, Italy 2007

[6] A. Patel, "Amit's Game Programming Site," http://www-cs-students.stanford.edu/~amitp/gameprog.html, Accessed May, 2007

[7] N. Sabanovic, "Planning of One Part Movement," *Project EACF05_00268*, Istanbul, Turkey, December 2005.

[8] V. I. Utkin, J. Guldner, J.Shi – Sliding Mode Control in Electromechanical Systems *CRC Press*, 1999.

[9] M. Bastan, "Visual Servoing of Mobile Robots Using Potential Fields," Masters Thesis, Sabanci University, Istanbul, 2004.

[10] M. Sitti, "Survey of Nanomanipulation Systems," *IEEE Nanotechnology Conference*, vol., no.pp.75-80, Nov. 2001

[11] A. Menciassi, A. Eisinberg, I. Izzo, P. Dario, “From "macro" to "micro" manipulation: models and experiments,” *Mechatronics, IEEE/ASME Transaction* , vol.9, no.2pp. 311- 320, June 2004

[12] M. Sitti, and H. Hashimoto, “Controlled pushing of nanoparticles: modeling and experiments,” *Mechatronics, IEEE/ASME Transactions* on , vol.5, no.2pp.199-211, Jun 2000

[13] D. Roth, “Vision Based Robot Navigation,” Masters Thesis, Massachusetts Institute of Technology, Cambridge, USA, 2004.

[14] K. Lynch, “Nonprehensile Robotic Manipulation: Controlability and Planning,” PhD Thesis, The Robotic Institute, Carnegie Mellon University, USA, 1996.

[15] “A* algorithm implementation in C#,” (updated May 2006), www.codeproject.com , Accessed June, 2007

[16] E.D. Kunt, “Design and Realization of a Microassembly Workstation,” Masters Thesis, Sabancı University, Istanbul, Turkey, 2006.

[17] J. Kuffner, S. Kagami, K.Nishiwaki, M. Inaba, and H. Inoue, “Online Footstep Planning for Humanoid Robots”, IEEE Int’l Conf. On Robotics and Automation (ICRA ‘2003), September, 2003.

[18] P. Lester, (updated July, 2005) “A* Pathfinding for Beginners,” http://www.policyalmanac.org/games/aStarTutorial.htm, Accessed June, 2007

[19] A. Malima, E. Demirok, and A. Sabanovic, “*Experimental Investigation of High Accuracy Motion Controllers*”, TOK07 (Turkish Automatic Control), Istanbul, Turkey. (in Turkish)

[20] A. Malima, and A. Sabanovic, “*Motion Planning and Assembly for Microassembly Workstation*,” Intelligent and Systems Control, ‘ISC’ 2007 Cambridge, USA.