AUTONOMOUS ROBOTICS

Mubbasir. T. Kapadia, Adil Lakahani and Rohit Jhangiani.
Under the guidance of: Professor M.V Deshpande and Professor Kiran Bhoumik.

Abstract – The aim of the project, to wirelessly inter-connect multiple autonomous devices via a computer which will serve as an interface. The devices will use multiple sensors to scan the external environment to procure data of the unknown region. Data sent to the central server will be processed to generate an accurate cartographic depiction of the territory traversed that will span all three dimensions.

Index Terms—Autonomous Robotics, Collaborative Thinking, Map Traversal, Neural Networking, Artificial Intelligence, Swarm Intelligence.

INTRODUCTION

This document gives a detailed analysis of the procedure that was followed in the implementation of the project titled Autonomous Robotics. A detailed analysis of each and every aspect of our project will be followed by an overview of the various application areas where our creation finds use.

1) Source of Inspiration

There is currently a large amount of research work going in the vast fields of autonomous robotics. Albeit in its nascent stages, research groups like BARC and TIFR have shown interest in the fields of Collaborative Thinking. The ultimate aim: to have multiple autonomous devices to work together in a collaborative manner to perform a complex task in the most efficient manner possible. These devices will be equipped with limited processing capabilities and will thus singularly not be able to successfully reach the end result. Thus having multiple devices, each performing a separate function, intermediate results can be integrated to achieve a final solution.

2) Wire-frame model

Our basic model is segregated into the following modules and functionality:

a. Creation of an external device using embedded technology.
b. The external device will be equipped with a control unit in the form of a microcontroller.
c. Equipping the devices with sensors to respond to its external environment.
d. Creation of multiple devices with varying processing power and functionality as per requirements.
e. Inter-connection of multiple devices using the computer which serves as an interface.
f. Wireless connectivity to remove the constraints of wires.
g. Creation of an autonomous device with self navigation capabilities thus implementing the essence of Artificial Intelligence.
h. The external devices will be aware of the relative progress of each other thus allowing Collaborative Thinking.
i. Implementation of a Neural Network with back propagation to facilitate response to a dynamically changing environment.
j. Information procured by the devices by means of the sensors will be sent to the central server. Data processing at the server will generate an accurate cartographic depiction of the territory traversed.

I. DEVICE CREATION

The first step is to create a basic model of a vehicle that has the capability of rotational and translational motion with considerable accuracy. Our external device would, as per command signals given by a control unit or a computer be able to navigate a path laden with obstacles with sufficient ease.

The external device has the following modules:

1. Actuators
2. Driving Mechanism
3. Control Unit
4. Sensors
5. Power Supply

1) Actuators

An indispensable part of our device is the actuator which
will provide the device with the ability to move. These are the motors that will provide the angular velocity to facilitate the wheels to rotate about their axis. Motors are available with speeds ranging from 30 rpm to 1000 rpm. However, angular velocity is not the only governing factor in the choice of the motor as per the following equation:

\[ \text{Power} = \text{Torque} \times \text{Angular Velocity} \]

As can be observed, a fine balance has to be achieved between the angular velocity and the torque of the machine.

We have chosen to make use of D.C Motors and its enhancement Servo Motors which has error correction facility by implementing feedback control. We will also be taking into consideration Stepper Motors due to the accuracy achieved in rotational motion.

2) Driving Mechanism

This is the framework of the entire chassis of the vehicle which will govern the manner in which it performs rotational and translational motion. After doing sufficient research and brainstorming we have decided to implement a Differential system with certain enhancements to suit our needs.

![Forward Translational Motion](image1)

**Fig: Forward Translational Motion**

![Backward Translational Motion](image2)

**Fig: Backward Translational Motion**

![Anti-clockwise Rotation](image3)

**Fig: Anti-clockwise Rotation**

![Clockwise Rotation](image4)

**Fig: Clockwise Rotation**

3) Control Unit

This is the central unit or the microcontroller present on the device itself which will govern the speed and direction of motion. In addition, it will process all data acquired by means of sensors or serve as its intermediate repository.

We will be implementing a control unit with feedback control to compensate the error due to physical constraints such as friction and other external factors that cannot be foreseen.

![Control Unit with Feedback](image5)

**Fig: Control Unit with Feedback**

4) Sensors

We will be extensively making use of wide variety of sensors such as Infra-red sensors, CCID devices light emitting diodes which will serve as our link to the outside world and facilitate interaction with the external environment. Use of image processing will be utilized to process the data of the surrounding environment.

5) Power Supply

The choice for power supply will be 12V portable batteries due to the wired constraints placed on all other alternatives. Solar powered devices could be considered as a possible upgradation of the project.

II. COLLABORATIVE THINKING

1) Swarm Intelligence

Swarm intelligence (SI) is an artificial intelligence technique based around the study of collective behavior in decentralized, self-organized systems. The expression "swarm intelligence" was introduced by Beni & Wang in 1989, in the context of cellular robotic systems.

SI systems are typically made up of a population of simple agents interacting locally with one another and with their environment. Although there is normally no centralized control structure dictating how individual agents should behave, local interactions between such agents often lead to the emergence of global behavior. Examples of systems like this can be found in nature, including ant colonies, bird flocking, animal herding, bacteria molding and fish schooling.

2) Our System – A centralized system

The original concept of swarm intelligence based upon the real life examples of ant colonies lack the centralized
intelligence mandatory in integration of the work carried out by each unit.

2) Two devices – Without collaborative thinking

In this scenario, multiple devices (two) exist but are not inter-connected to one another via a central server. Hence, there exists no collaboration between devices as each device is unaware of the other’s whereabouts.

As shown in the figures above, decentralization between multitudes of devices would necessitate an ad-hoc communication strategy between each and every device. A centralized approach would only mandate communication between the external device and the server. Inter-communication between devices would take place via the computer serving as an interface. The computer would store information regarding each and every device. Thus, each device would be aware of the relative position of every other device.

III. PATH TRAVERSAL

The aim: to efficiently traverse the entire map using multiple devices in the most efficient manner using collaborative thinking.

1) Map traversal using one device

When there is only one device present on the map, collaborative thinking does not come into the picture. Thus, the single device traverses each and every free area of the map.

3) Collaborative Thinking

Here, the devices are aware of each other’s relative positions and thus do not traverse the area already scanned by another device. Thus, the efficiency of traversal is greatly improved.

IV. PATH TRAVERSAL ALGORITHM

The algorithm to implement path traversal for the devices forms an essential ingredient in order for the
project to succeed. There are a large number of algorithms that can be implemented in path traversal. These algorithms differ only in the strategy used for exploring the four possible directions from the current point. In all cases movement in a direction is disallowed if the neighboring point in that direction contains an obstacle or if that neighboring point has already been visited. All algorithms terminate when either the goal point is found, in which case success is reported, or when all reachable points have been explored without finding the goal point. In the case of map generation there is no goal case and algorithm terminates when the entire area is scanned.

The following algorithms and their variations can be implemented in path traversal:

1) **Fixed Order**

   This algorithm uses a fixed order in which it tries one of the four directions for traversal. The order of priority from highest to lowest is:

   I. North  
   II. East  
   III. South  
   IV. West

2) **Maintain Direction**

   This algorithm gives greatest priority to the direction in which the device is currently traveling. Using this algorithm, turning left or right would be minimized which could potentially slow a device. If the direction of traversal is blocked then priority is assigned as follows:

   I. Left  
   II. Right  
   III. Backwards

3) **Towards Goal**

   This algorithm has prior knowledge of the final coordinate and must only find the path to traverse in reaching that point on the map. It will always give first priority to the vertical or horizontal direction that is closest to the straight-line direction from the current point to the goal. It will give second priority to the direction second closest to the straight-line direction. The third priority should be the opposite of the second direction and the fourth priority should be the opposite of the first direction. This algorithm ignores the current direction of travel.

4) **Towards Goal – Maintaining direction**

   This algorithm combines the best of the previous two strategies. It gives first priority to maintaining the direction which is currently being traveled. This minimizes turning left and right. The second priority is given to the direction that is closest to the goal state. The third priority is given to the other direction that is next closest to the goal state. The fourth priority is given to the last direction.

   ![Diagram](current_direction_diagram.png)

The above algorithms are designed to find a particular location on a map. Also, they can cater for only single device and would not provide optimal solution for multiple devices implementing collaborative thinking. Hence, we have devised the following variations keeping in mind the following factors:

1. There exist multiple devices traversing the map simultaneously in a collaborative manner.
2. There is no particular goal state: No pot of gold exists.
3. There exists an anti-goal state: The location of the other devices.
4. Each and every free area of the map has to be traversed.
5. The locations of the devices are constantly changing.

5) **Away from Goal**

   This is a variation of the above strategy in which a coordinate is specified and map traversal takes place with a best effort to stay away from that particular point on the map. This form of map traversal can be particularly useful in case of multiple devices traversing the map in a collaborative manner. The starting coordinates of the other devices would be known to each device and the algorithm would employ a best effort strategy to stay away from the other devices.

6) **Dynamic Away from Goal**

   This is a proposed enhancement of the static strategy. In the case of static away from goal, only the starting coordinates of the devices would be taken into consideration. However, this would not result in optimal path traversal as the coordinates of the devices would
continually change.

In case of dynamic away from goal strategy, the relative positions of every other device are known to each device. Based upon this knowledge, the device computes a direction of traversal which causes it to move away from the devices.

Consider the case of three devices A, B and C. The priority given to order of movement is as follows for device A:

I. Move away from B and C
II. Move away from B
III. Move away from C
IV. Move towards B and C
V. Move to location already traversed
VI. Move backwards

V. GENERALIZED ALGORITHM

Each maze traversal algorithm should work in the same manner except for the strategy used on visiting neighboring points while searching for the goal. The outline of the basic algorithm is as follows:

I. Initialize stack S to empty state by assigning S.TOP to -1
II. Push coordinates of starting point into stack S
III. While S.TOP != -1
   a. Pop element from stack into variable A
   b. If A is GOAL state
      i. Print path
      ii. Return
   c. Set A to current location
   d. Push A back into stack
   e. Find neighbors of A
      i. B = A.NORTH
      ii. C = A.EAST
      iii. D = A.WEST
      iv. E = A.SOUTH
   f. Push neighbors onto stack if they exist
   g. If no neighbors pushed onto stack then A is said to be DEAD END.
   h. Continue popping from the stack until new location reached.
   i. Push new location onto stack
   j. Continue
IV. Stop

VI. MAP SIMULATION IN C

1) Map Definition

The input for your program will be the map of a maze contained in a file named mazemap in the current directory. The file contains the maze where the character `#' represents walls and obstacles in the maze, a space ` ' represents a free point, `<' represents the starting point and '>' the goal. These definitions are captured in the following:

```c
#define WALL '#'
#define FREE ' '
#define START '<'
#define GOAL '>'
```

As shown in the following maze example, you may assume that a maze is always completely surrounded by a wall so you will not have to make any special checks for boundaries.

```
##########
#        #
#  #> # #####
#  #  #     ###
#  #     ##   #
#  # ##########
#  ###   #
## # # # #
#<       #
##########
```

2) Point Definition

Traversing a maze requires movement between points. In this assignment a point is defined as an (row,column) coordinate. Because some of the algorithms also need to use the current direction of travel you should define an enumerated type for the direction with the resulting definition for a point as follows:

```c
enum directions {north, south, east, west};
struct Point {
  int row;     /* row coordinate */
  int col;     /* column coordinate */
  enum directions dir; /* direction of travel in reaching this point */
};
```

3) Map Abstraction
You will need to create a \textit{Map} abstract data type (as a C++ class) to store the maze. This abstraction should be implemented using a two dimensional array where the maximum vertical dimension of a maze is 20 (number of rows) and the maximum horizontal dimension of the maze is 80 (number of columns). Important: the upper left corner of the grid is considered to be coordinate (0,0). The abstraction can be defined as:

```cpp
#define MAXROWSIZE 20
#define MAXCOLSIZE 80

class Map {
private:
    int maxrow;  /* actual maximum row dimension */
    int maxcol;  /* actual maximum col dimension */
    char grid[MAXROWSIZE][MAXCOLSIZE];
/* the grid itself */
public:
    Init(Point &ptStart, Point &ptGoal);
    SetPoint(Point pt, char ch);
    char GetPoint(Point pt);
    Print();
};
```

You will need the following methods to access this data structure. Your maze traversal algorithms should call these routines to access the map.

```cpp
/*
 * Init - initialize the map from the input file and find the start and goal points.
 */
Map::Init(Point &ptStart, Point &ptGoal)
/*
 * SetPoint - set the value of the map at the given point to the given character
 */
Map::SetPoint(Point pt, char ch)
/*
 * GetPoint - return the value of the map at the given point
 */
char Map::GetPoint(Point pt)
/*
 * Print - print the given map
 */
Map::Print()
```

The initial direction for the starting point is always assumed to be north using the following declaration. This value should be set in the \textit{Map::Init()} routine.

```cpp
#define INITDIR north
/* initial direction of travel */
```

The main program should define a variable of type \textit{Map} class and invoke the \textit{Init()} method to initially read and create the internal representation of the maze. \textit{Initiate()} should first set the entire grid to the value FREE before reading in the maze. The routine should also determine the actual size (maximum number of columns and rows) in the particular maze for later printing.

In addition to reading in and printing the map, the \textit{Init()} routine will also need to verify that the map is valid by containing both a start point and a goal point. If the map is valid, the routine should return a zero, otherwise return a negative value to indicate an error. The location of the starting and goal point also need to be returned. These are included as call-by-reference parameters to \textit{Init()}.

VII. Applications

1) Unmanned Military vehicles

Swarm Intelligence-based techniques can be used in a number of applications. The U.S. military is investigating swarm techniques for controlling unmanned vehicles. With the rise of Artificial Intelligence in Military applications, multiple unmanned vehicles can be made to work in a collaborative manner to perform planned group attacks.

For e.g. flying aircrafts, terrain vehicles as well as warships can all be coordinated to perform a planned attack on the enemy base without the need for loss of life.

2) Scouts

Autonomous devices could also be sent as scouts to scan hostile territory for the presence of enemies. Here multiple devices could be used which are extremely small and can go undetected. One particular robot would be provided with sensing capabilities while the other would be provided with a defense mechanism and would simply follow the scout.

3) Map Generation

The primary purpose of the project is to generate an accurate cartographic depiction of the territory traversed by the multiple devices. Initially the map would be designed to cater for two dimensions but would later be enhanced to provide 3-dimensional mapping. Thus, the map would provide information regarding the topographical features and geographical features such as inclination and declination of region.
4) Space Research

NASA is also conducting research in the field of collaborative thinking. They are designing a swarm of robots modeled on ants that are of extremely small size and have limited processing power. These ‘ant’ robots will be placed on different points in unknown lands such as the unmapped regions of Planet Mars. These ant robots would be connected to a central server computer and would work collaboratively to scan the region. Information sent to the server would be processed to generate an accurate cartographic depiction of the place. The ant robots could thus scan the region for other things such as fossil remains, presence of life or water.

5) Pipe reconstruction

Companies in India such as L&T are also employing the use of multiple autonomous devices in the repair of pipes in underground regions beyond the reach of man. A scanner robot first scans the area for possible pipe leaks. Upon identification of a leak, it sends a signal to another device which is capable of performing repairs. The small devices are thus each given a particular task which they can perform successfully.

6) Mining

These autonomous devices can also be used in the detection of mines, a job that is fraught with danger. Here, extremely small light weight robots could be used and each be given a separate task. For instance, a scanner robot would be coupled with sensors that allow it to scan the area for the presence of mines. Upon detection of a mine, it would send a signal via the server to a robot which has the capability to disarm the mine. Multiple scanner robots could be utilized in the efficient scanning of the territory.

7) Mountainous regions, Forests and Deserts

These devices could be transformed as all terrain vehicles so that they could traverse difficult mountainous terrain which could be beyond the limits of humans.

Similarly they can be used for traversing dense forest conditions or difficult desert conditions where they could function as follows:

I. Searching for human existence.
II. Searching for food.
III. In search for dead bodies.

8) Medicine

Devices designed at micro and nano levels could be utilized in the field of medicine. Here, multiple extremely small devices could be inserted into various parts of the human body for extraction of external unwanted substances or for analysis of internal organs.

VIII. CONCLUSION AND FUTURE WORK

The technical paper gives a detailed analysis of the various aspects of our final year project titled, ‘Autonomous Robotics’. It gives an outline of the following areas of relevance as concerned with our project:

I. Robotics
II. Collaborative Thinking
III. Path Traversal
IV. Map Generation

In the future, we as a team aim to carry out an extensive study of Neural Networks and incorporate it into our project by integrating neural networks into our autonomous control system.

IX. ACKNOWLEDGMENT

We would like to take this opportunity to thank the Computer Department at D.J Sanghvi College of Engineering for providing us with the necessary resources needed for the successful implementation of the project.

We would like to thank Ms. Kiran Bhoumik who is our co-guide for our final year B.E Project. I would also like to express my gratitude to Professor M.V Deshpande, who in his capacity as project guide and head of department, provided us with the guidance and helping hand needed in the successful deployment of the project.

X. APPENDIX A – EXISTING PROJECTS

Free-ranging military vehicles

There are 3 clusters of activity relating to free-ranging off-road cars. All these projects are military-oriented.

1. US military DARPA Grand Challenge

The US Department of Defense announced on the July 30, 2002 a “Grand Challenge”, for US-based teams to produce a vehicle that can autonomously navigate and reach a target in the desert of the southwestern USA.

In March 2004, the first competition was held, for a prize-money of $1 million. Not one of the 25 entrants completed the course. However, in October 2005 five different teams completed the 135-mile (217 km) course, and the Stanford University team won the $2 million prize.
Following the 2004 failure, in which several cars were distracted by the "race" to the detriment of basic technology that would allow for actual completion, the 2005 teams were focused on the challenge at hand, and did not seek to develop generic solutions, or a particularly speedy car. By and large, the sensors used were stabilized in order to avoid the vibration of desert driving. The sensors were based on Visual, Radar, and laser technologies. The navigational course was pre-programmed, and the micro-navigation and obstacle avoidance were handled by on-board computers - many of the entrants used 8 or more computers to manage the car. Though the vehicles were equipped to avoid collision, they did not have any notion of rules-of-the road - but simply regarded each other as moving obstacles.

2. European Land-Robot Trial (ELROB)

Not to be outdone by the USA, the German Dept. of Defense announced an event similar to the DARPA Grand Challenge, held in May 2006. The event included both desert-like scenarios like in the USA, and also urban scenarios in which the vehicle will explore streets and buildings. In August 2007 a civilian version of the event will be held in Switzerland.

3. The Israeli Military-Industrial Complex

As a follow-up from its success with Unmanned Combat Air Vehicles, and following the construction of the Israeli West Bank barrier there has been significant interest in developing a fully-automated border-patrol vehicle. Two projects, by Elbit Systems and Israel Aircraft Industries are both based on the locally-produced Armored "Tomcar" and have the specific purpose of patrolling barrier fences against intrusions.

ARGO

ARGO is an Italian project (1996-2001) to allow a car to follow the normal (painted) lane marks in an unmodified highway. The culmination of the project was a voyage of 2,000 km during 6 days in the motorways of northern Italy, with an average speed of 90 km/h. 94% of the time the car was in fully automatic mode, with the longest automatic stretch being 54 km long.

The ARGO vehicle, a modified Lancia Thema, had only two Black-and-White video cameras on board, and used stereoscopic vision algorithms to understand its environment. This is in stark contrast to the "laser, radar - whatever you need" approach taken by other efforts in the field.

The project was run by the universities of Parma and Pavia, coordinated by Alberto Broggi, and financed by the Italian government.

XI. References

[1] Wikipedia, Swarm Intelligence