A REPORT

ON

EXPERIMENTAL STUDY ON CONCURRENT AND SOCIAL LEARNING ALGORITHMS IN ROBOT TEAMS

BY

RAGHAVENDER SAHDEV, BITS PILANI, INDIA

UNDER THE SUPERVISION OF

Prof. Reza Emami

Institute of Aerospace Studies

Engineering Science Department

UNIVERSITY OF TORONTO, TORONTO, CANADA

ST. GEORGE CAMPUS

August, 2014.

ACKNOWLEDGEMENT

A report always requires the goodwill, encouragement, guidance and support of many people. The all-round aspect thinking that an engineer has to have can hardly be gained through books and classes. The exposure to industries, learning and fulfilling their requirements make us feel more confident about our knowledge, and such a learning process is very motivating to keep learning more. Right from design aspects, protection schemes, calibration and maintenance to troubleshooting are part of our knowledge.

All this would not have been possible without the constant help of my supervisor Professor Dr. Reza Emami, Institute for Aerospace Studies, University of Toronto throughout the project. I am deeply indebted to him for giving me the opportunity to work under his supervision.

I would also like to thank Justin Gerard for his valuable inputs he kept pouring. I would also like to thank my colleague Jai Bansal for his valuable contribution and suggestions throughout the course of this project.

CONTENTS

INTRODUCTION

This project is a part of previous research done on algorithms developed for Concurrent Individual and Social Learning in robot teams. This report focuses on the experimental study of implementing concurrent and social learning in robot teams. This has been virtually tested on software (Matlab) through a simulation. This report focuses on preparing the test-bed and environmental conditions for the smooth execution of the algorithms and testing those algorithms practically through hardware.

SOFTWARE USED

Following Software have been used in the experimental study –

- Open CV 249 – Open CV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. The library is cross platform. It focuses mainly on real-time image / video processing. Open CV libraries in C++ have been used for the purpose of video processing due to its fast execution.
- Matlab – Matlab is the entity which is responsible for sending various commands from the XBee radio module attached to the central server (here the laptop) to each of the 12 rovers. These signals are sent after running machine learning algorithms (Q Learning and L Alliance) in the simulation, the results of which are commands sent to the rovers.
- Arduino IDE – This is the software being used to upload the code to the arduino microcontroller board (Arduino Mega 2560).
- Digi XCTU – XCTU is a free multi-platform application designed to enable developers to interact with Digi RF modules through a simple to use graphical interface. It includes new tool that make it easy to set-up, configure and test XBee RF modules. Here It is being used for testing the communication between the server XBee radio module and the XBee attached on each of the rovers.

PHASE 0

HARDWARE PREPARATION

PREPARATION AND HARDWARE TESTING

This phase involves the testing of various parts of the rover. This phase is required so that the requisite hardware is prepared for the smooth running of the phases that follow this phase. The available code was tested in this phase in order to check that all the components of the rovers function properly.

SONAR Sensor – The SONAR Sensor is being used for obstacle avoidance. The SONAR Sensor has a range of about 2 meters maximum. SONAR senses an obstacle upto a distance of 1

meter accurately beyond which it gives it fails to detect. The sonar sensor was tested for the 12 rovers and found to be working for all the rovers.

Servo motors – The Servo motors attached to the sonar sensors and the camera was tested. This is required for obstacle detection and avoidance in the left and right side of the rover. The sensor calculates the distance between the rover and the obstacle and accordingly makes the servo motor rotate and drives the rover in a particular direction. The servo motors were found to be working normally for all the rovers

Motors – Motors were tested for the rovers to ensure that all of them are in proper condition. The connections between the motors and motor driver were checked to ensure proper connections. 2 of the robots had loose connections with their motor driver which were fixed.

Shaft Encoders – Shaft Encoders were tested for the rovers. They are being used for odometry. The rovers were made to rotate 1 complete rotation as per 10 ticks of the shaft encoder light sensor. 1 complete rotation corresponds to 27.1 centimetres (Circumference of the wheel) distance on ground. So depending on the number of ticks of the shaft encoder we can estimate the distance moved by the rover.

Distance Moved = (Number of ticks) * 27.1 / 10 centimetres

Shaft Encoders are being used for Localization purposes. Localization is the problem of estimating the place of the robot relative to a map. In other words robot has to answer the question, *Where am I?* The rover will poll the Microsoft Kinect sensor and ask about its location and then check with the calculated location by the shaft encoder. It will then remove any errors that may have occurred due to slipping of the wheels. This process would be repeated after regular intervals.

XBees – Xbees are radio modules used for wireless radio communications. They are required for the wireless communication to take place between the rovers and the main server XBee attached to the computer. Signals were broadcasted from the Server XBee to the rovers and depending on the robot ID that was sent in the packet, the corresponding robot performs the action (like move forward 40 cms, rotate servo at 30 degrees, etc). XBee radios are required by rovers to communicate with each other to ensure collective and cooperative behaviour.

Ten rovers were assembled and 2 more were required to be assembled. It was found that one arduino board was non-functional and two XBees were required for two rovers. Accordingly one rover was assembled and the components required for the last rover were ordered. One rover was assembled and connections were tested.

Bumpers – Long thin aluminium strips are attached to a micro-switch which triggers an interrupt on making contact with an obstacle. There are 4 bumpers, two on the rear and

two on the front side. The Older design had a lot of flaws. Most of the bumpers on the rear side were making contact with the wheels so accordingly changes were made in the new design. The rear side bumpers were bent so that they do not make contact with the wheels.

At the end of this phase proper functioning of the following components of the rovers was ensured:

- Motors Left / Right
- Motor Drivers
- Battery (Charged / Discharged)
- Servo Motors attached to the Sonar and Camera
- Sonar Sensors
- Bumpers (4 on the rear and front)
- Switches (DPDT and SPST)
- XBees for radio communication
- Shaft Encoders attached to the Motors
- Wiring connections between all components

BATTERY BACKUP

The batteries of the rovers were tested and it was observed on an average a battery runs for 40-45 minutes. One of the rovers (robot 5) has its battery drained off. It takes approximately 3-4 hours for the battery to be fully charged.

Physical Condition of the robots in August, 2014

Robot 2,3,4,6,8,9,10,11 work perfectly fine

Robot 5 has its battery drained, gives a backup of 10-15 minutes only when fully charged. The battery has to be replaced

Robot 7 needs to have the switch connections replaced

Robot 12 needs to be assembled

ON BOARD CAMERA

Existing camera module is OV7670 + AL 422. This camera has a FIFO Buffer. When transmitting a frame, the FIFO buffer holds the data and slowly passes it to the arduino because arduino cannot handle such big data at once. So a buffer on the camera module solves the purpose.

Live video streaming is not required in our experiment as the robot should only know about its vicinity when it encounters some obstruction in its path. So it was decided that taking a snapshot would solve the purpose. It was then decided that every robot would be

multiplexed and the robot could be selected based on the robot ID and accordingly the corresponding robot would take a snapshot.

The problem to be solved is that the rover should take a snapshot and then wirelessly transmit that image serially to the laptop for processing. Following methods were found apt for the problem to be solved:

1. Using Arducam shield with the existing camera module.
2. Wireless camera with Arduino and the Adafruit CC3000 breakout board
3. Using some other micro-controller like raspberry pi, mbed.

After considering various options depending on their feasibility (cost, efficiency) we came up to the conclusion that Arducam Shield would be the best option as it could be stacked on top of the already existing microcontroller. The arducam shield would be stacked on top of the XBee shield.

As of now the arducam shield that was ordered can not accommodate an XBee module with it. A separate arduino board is required if we plan to use the Arducam Shield.

PHASE 1

MAPPING

Mapping refers to creation of a map of the entire area being imaged and storing it in some form. Here our target is to map an area of size 10*10 sq. metres. Mapping is required to store the coordinates of items, obstacles and the target zone. This is required because the simulation assumes in the beginning that prior information of the items, obstacles and target zone positions is already there and is used in the simulation.

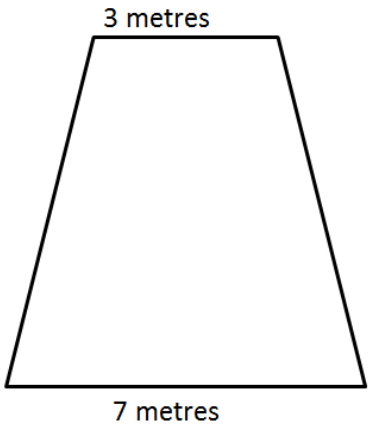
SENSOR SELECTION

MICROSOFT KINECT SENSOR

Initially the Microsoft Kinect sensor for Xbox360 was being used to capture the information about the various details of the area. Kinect was initially placed over head to capture a top view of the area. Kinect was placed at a height of 2.2 metres vertically above ground and it could capture a top view area of only 2 m * 2.5 m. Increasing the height to 3 metres did not significantly increase the projected area.

Kinect was then placed at a height of 2.4 metres at angle of approximately 40 degrees, it captures a isosceles trapezium projected on ground of base lengths 3 metres and 7 metres. This is the maximum area that can be captured by a single kinect sensor. Kinect sensor has a

low field of view. It has a 43 degrees Vertical Field of View (FOV) and 57 degree Horizontal field of view.



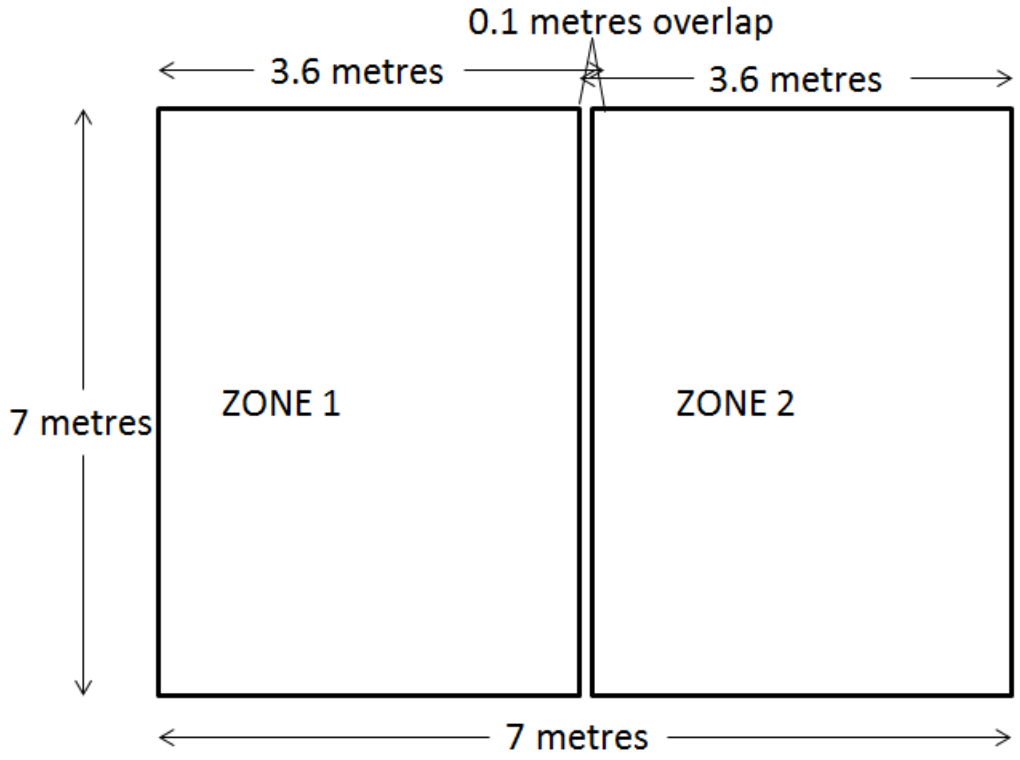
Maximum Area Mapped by 1 Kinect

Kinect mapped this area when placed at an angle and the view was not a top view, so a transformation had to be done which added to the computational complexity of the video processing task in Matlab. The error in x direction was approximately 30 centimetres which was not at all acceptable. The inability of kinect to capture a wide area on ground and a top view and depth sensor not being used are some of the factors which lead to the need of another sensor for Mapping.

A sensor with a high field of view was needed. The depth sensor of the kinect has a range of 4 metres. In our experiment we do not need a depth sensor because all we need is just the image / video data. So it was decided to go with the option of a web camera with a high field of view.

WEBCAM

Various webcams were searched and the webcam (Genius Ultra Wide Angle PC Conference Widecam F100 Full HD Webcam) was selected. This webcam has a field of view of 120 degrees. It has a FOV of 120 degree in the horizontal direction but has a lesser FOV in the vertical direction. The advantages of using the webcam over the kinect include its low cost, higher FOV and small size. The webcam when placed at a height of 2.2 meters captured a projected area of 3.6 by 7 sq. metres. The Web cam was fish eyed due to its high FOV. There was some curviness near the edges of the camera. One camera captured an area of 3.6 m by 7 m. So it was decided to use 2 of these cameras and together they could capture an area of 7 by 7 sq. metres.



Full view captured by the 2 cameras

As can be seen from the above picture the area has been divided into 2 zones which correspond to the view from each of the 2 cameras and accordingly positions of items and target zone have been mapped in the code. One major assumption has been made in the software code written for mapping – all the items should either belong to one zone or the other, it cannot belong to both the zones implying lying on the intersection of the 2 zones. The view that the camera gives is fish eyed which is discussed in the latter part of the report.

The only thing that is being done by the camera is video processing. So for mapping various items, obstacles and target zone, the camera should be able to tell the coordinates of all the items, obstacles and the target zone.

SOFTWARE PLATFORM

The Camera was set to capture a 1280 * 720 resolution video. Initially Matlab was being used for mapping purposes. Matlab being an interpreted language made the task of video processing computationally expensive. It was seen that Matlab gives lot of lag when streaming a video stream of such high resolution. Certain localization tests were run using blob analysis to find connected components in the video using the Matlab. It was seen that Matlab gave a lot of lag while processing the video. Hence it was concluded that Matlab should not be used for Video Processing a live stream. So another external tool was required for processing the video processing algorithm.

It was decided that Open CV library would be used for the video processing and the coding platform to be used would be C++. Open CV library offers a wide variety of inbuilt codes which can directly be used for video and image processing. So the computational load of video processing would be completely taken off from Matlab and Open CV would run the required code for Mapping and Localization. Open CV would store the coordinates of items, obstacles and target zone in a notepad file. The only job of Matlab would be to read those coordinates from the notepad file which it could use for running the Simulation. This would reduce the time taken for localization of rovers.

LOCALIZATION

Localization refers to the process in which the robot is made aware about its location. By localization the robot knows at each point of time about its coordinates in the 2 D space and its orientation. Here by localization we mean the robot can answer at each point of time – *"Where am I?".*

Localization was done for the rovers to know about their current position at each point of time. Localization could be done in 2 ways either by using the overhead cameras or the local shaft encoders mounted with the motors of the rovers.

USING OVERHEAD CAMERAS

Overhead cameras capture the image of the scene frame by frame and process each frame step by step. The algorithm being used for finding rovers in the video is simple blob analysis. Initially each RGB frame is converted into a HSV image, the frame is then threshold by values such that only the pixels belonging between the threshold values retain their content others are removed.

The camera's view is fish eyed at the borders due to the high field of view of the camera. This is prominent in the horizontal direction much more than in the vertical direction due to high FOV in the horizontal. A direct one to one linear mapping is done. One camera captures a view of size 1280 * 720 pixels resolution. In the horizontal (x) direction this is equivalent to 700 centimetres and in the vertical direction 720 pixels correspond to 360 centimetres. So a linear relation is used between pixels –

1280 pixels = 700 centimetres (in x direction)

720 pixels = 360 centimetres (in y direction)

'a' amount of pixels in x direction corresponds to a * (700/1280) centimetres

'b' amount of pixels in y direction corresponds to b * (360/720) centimetres

A point at (a,b) has coordinates {a*(700/1280), b*(360/720)} centimetres on ground.

If the particular item or the target zone belongs to zone 2, coordinates of the x direction remain unchanged. In the y direction 720 is added to the pixel value and then the coordinates are calculated as per the following relation

'b' amount of pixels in y direction correspond to b * (720/1440) centimetres.

Each of the rovers has mounted wooden supports on top and a coloured paper is attached to it. Now for points immediately below the camera the localization error in the x direction is within the acceptable limits, but as we move farther away this error rises by a constant amount based on the distance from the central point. To correct this error the area imaged by the camera has been divided into 10 zones in the x direction with 5 on either side of the camera and a constant is added to the x coordinate of the rover depending on the zone to which the point belongs to. Following table shows this correction:

Distance from centre in x direction (in pixels)	Value from the camera	Corrected Value
0 – 150	X	X + 40
150 – 300	X	X + 35
300 – 400	X	X + 20
400 – 500	X	X + 10
500 – 640	X	X
640 – 780	X	X
780 – 880	X	X – 10
880 – 980	X	X – 20
980 – 1130	X	X – 35
1130 – 1280	X	X – 40

These correction terms in the x direction have been determined experimentally by taking readings of x coordinates of the rovers when placed at different positions.

LOCALIZATION ERROR

Localization error is the error in measuring the position of the rover. It is defined as the absolute difference between the actual position and the calculated position by the overhead camera.

The localization error was calculated in the x and y direction. A total of 60 readings were taken and these correction terms did not allow the localization error to exceed the acceptable limit (maximum 10 centimetres) in the x direction.

In the y direction the localization error was also under the acceptable limit. It never exceeded more than 10 centimetres. So in total the robots error in localization was 10 centimetres both in x and y direction.

ORIENTATION OF THE ROVERS

Orientation of the rover is required to be known in order for the robot to know which direction is it pointing to. Orientation of the robot can be calculated by the following means

- By taking the frame difference between the consecutive frames
- By marking a marker at some distance from the centroid of the existing coloured paper on top of the robot.
- By calculating the difference in ticks of the shaft encoder

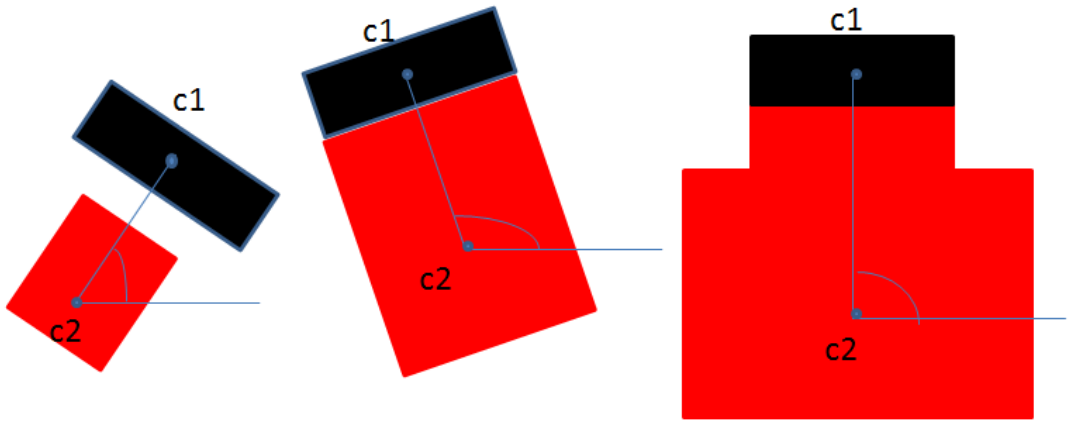
Option 1 was rejected because it would not give accurate results when the robot simply turns on its spot or does not move at all.

Option 3 could not be relied upon alone as the error in shaft encoders always builds up with time due to dead reckoning. So it is not guaranteed to be accurate after a certain amount of time.

Option 2 was found to be the most reliable as it is not dependent on the history of the robot. It stores no information about the rover's coordinates / orientation prior to the current step. It is similar to clicking a snap shot of the scene and telling the rover about its orientation using only the data in that particular frame / snapshot.

A black coloured strip of chart paper was attached on top of each rover. The coordinates of the black strip were calculated and compared with the existing centroid of the robot. Knowing 2 points of a rover, the orientation could be easily calculated using the concept of the slope of a line.

$$\text{Orientation} = \tan^{-1}\left((y2 - y1) / (x2 - x1)\right)$$



C1 and c2 are the 2 centroids which will be used to calculate the orientation of the robot

The error in orientation of the rover using this method was approximately 15 degrees.

We detected both the blobs (new colour added + the initial colour on the rover). We have 2 points so we can get the slope of the line and hence the orientation of the rover.

USING LOCALIZED SHAFT ENCODERS

Another option we have with us is to use local shaft encoders which tell about the location and orientation of the robot based on the number of ticks. Odometry is being used for localization. Odometry is the use of data from motion sensors to estimate change in position over time. Odometry is used by robots to estimate (not determine) their position relative to a starting point. This method is sensitive to errors due to integration of velocity measurements over time to give estimates.

Rovers were made to go in a straight line for 5 meters; most of the rovers gave high errors in measurements initially. For odometry to be efficient and accurate, following should be ensured:

- Proper Hardware – Mechanically the motors and the chassis should be in a proper condition. The shafts of the motors should not be loose.
- Changing PWM – The speed of each motor should be defined separately.
- PID Controller - Refine the arduino code to handle PID (Proportional Integral Derivative) control using the shaft encoder.

After applying the above mentioned techniques to the rovers, the rovers were made to travel in a straight line for 600 centimetres. The rovers on an average deviated for about 40 centimetres from their line of path. The robots behaved quite accurately upto 300 centimetres with an average error of 10 centimetres.

Then square tests were done in a square of 3m by 3m. The robots gave an average error of 70-80 centimetres in this test. In Odometry the error gradually builds up and finally adds to a really high error. Due to dead reckoning the error always builds up in odometry.

At the end of the odometry phase, it was concluded that odometry alone cannot be used for localization purposes. If odometry has to be alone used then stepper motors should be used instead of the geared motors being used currently. The advantage of stepper motors over toy motors is that it is more accurate as its motion is discretized into steps. The disadvantage of using stepper motors is its cost as they are more expensive than the plastic geared toy motors being currently used.

USING ODOMETRY WITH THE OVERHEAD CAMERA

Odometry and the overhead camera can be used together for the purpose of localization of the robot. This is another proposed approach for the task of localization. This has not been implemented as of now. The robot can rely on its localized shaft encoders which are known to be accurate for approximately 3 metres. It takes a robot around 12-15 seconds to cover 3 metres. So after every 10 seconds each of the rovers can poll the camera and enquire about

its actual location and then check with its calculated position from the shaft encoder. The rover can then zero out the difference in the error. This way it reduces the dependence on the overhead cameras which it polls only once every 10 seconds. This method has not been implemented as of August 2014.

PHASE 2

IDENTIFICATION OF THE TYPES OF THE ROVERS, ITEMS AND TARGET ZONE

This is another major task to be achieved. None of the rovers by themselves know that they are strong or weak, fast or slow. They acquire this knowledge over a period of time when they face circumstances in which they learn about their type. We achieved this task in this phase with the help of overhead cameras.

We have 4 different types of robots.

- Strong and fast
- Strong and Slow
- Weak and Fast
- Weak and Slow

We have 2 different types of items

- Heavy item
- Light item

In total we have 12 robots with 3 robots in each category.

Once a robot has been identified, it approaches an item that it has been assigned to pick up according to the algorithms instructions. When the robot is near an item zone it asks Matlab about its type and then Matlab accordingly tells the robot to pick up the item from the item zone or to back off from the item. Now this phase focuses on the robots acquiring knowledge about them and the how the robots know about the item that is front of them.

We have 2 ways of identifying the rovers

- By Tracking each of the rovers
- By visually making the rovers different

BY TRACKING

Each of the rovers can be individually tracked based on the prior knowledge of where (the coordinates) each one started. This may be achieved by using an algorithm which relies on the information about the past of the rover. Each robot would be given ids from 1 to 12 and

then 4 groups would be formed 1-3, 4-6, 7-9 and 9-12 to classify the rovers into 4 different categories. A Kalman filter was used in matlab to achieve this but as matlab is very slow for processing a video, the idea was dropped. Kalman filter relies on the past history of the rover making it dependent on prior knowledge about its location. The second approach however is independent of history of the robot which makes us more inclined to using it.

BY VISUALLY MAKING THE ROVERS DIFFERENT

In each of the four categories mentioned above we have 3 robots. In total we have 12 robots. The overhead camera is being used for telling each robot about its type. Four colours have been assigned to the 4 categories of robots. Camera runs the video processing algorithm and stores the information of each robot in a test file. This is how each robot acquires knowledge about its own self.

Colour selection

We first convert the RGB colour space to an HSV colour space because it is more efficient to work in the HSV colour space than RGB. We require 4 different colours for the 4 types of rovers such that they can be distinguished easily from each other. We also need an additional colour for marking a strip / tail of the robots in order to calculate the orientation of the robot. Various colours (red, green, blue, black, pink, yellow, orange, brown, grey, cyan, etc.) were tried and finally we came up with the following four colours for the robots – orange, yellow, blue and green. Black was chosen to mark the tail of all the rovers for calculation of the orientation.

Making different shapes

Different shapes (circle, triangle, square, star) were created and tried to distinguish from the rest. At the points directly under the camera the shapes could be detected but as we moved a little further away from the central point of focus of the camera, the shapes appeared as distorted in the camera and could not be detected.

Finally, we have assigned four different colours to the 4 different categories of the rovers. In each category of robot, we have 3 robots which belong to the same type. (example all the three rovers are strong and fast). So it is required for us to be able to visually distinguish each of the three robots in a particular category. This has been achieved by having a rectangular coloured piece of paper of 3 different sizes. Now we can distinguish the 3 robots of a particular category by calculating the area of them and comparing them amongst each other.

Initially a simple A4 size paper was tested by printing various colours on it. The colours were selected based on their HSV values. So various colours were printed and tested for the HSV values. It was seen that orange, yellow, green, blue and black were easily distinguishable from each other.

So they were put on top of the rovers and tested. The paper was reflecting light incident on it. There are many overhead bright lights on the roof due to which the colour was not being detected properly at some spots. So to tackle this issue initially it was proposed to get rid of the overhead lights in order for the colour detection to be accurate at all points on the area being imaged. So 7 studying lamps were used in such a way that they did not point vertically on ground and were kept inclined at some angle facing the roof. This ensured proper detection of colours. Later we found a chart paper whose material was such that it did not reflect any incident light so the overhead lights did not pose a problem. We got various colours of the chart paper and each of them was tested and the results for the colour detection in presence of the overhead lights improved drastically. The overhead light need not be turned off for the experiment to be conducted. However in the corner a lamp is being used because it appears a bit dark in absence of the lamp.

IDENTIFYING THE ITEMS AND THE TARGET ZONE

Identifying the items was not a big challenge as they were big in size (40*40) centimetres. So they had clearly distinct blob areas whose blob area was much larger than the rovers. So practically we could choose any 2 different colours. We chose red for the light items and black for the heavy items. After choosing the colours their mapping was simple enough and the coordinates were stored in a text file after running the code for around 10 seconds. Mapping the target zone was also not a challenge as its blob area was more than 10 times the item zones' area (1.5 * 1.5 sq. metres). So after running the code for an initial 10 seconds the item zones coordinates and the target zone coordinates were stored in a file and fixed throughout the latter part of the experiment.

IDENTIFYING THE OBSTACLES

Initially obstacles were also to be mapped by the overhead cameras, but as this could be done by the localized bumper switches also the overhead cameras were not used. Also the ultimate goal of this experiment is be able to run without the aid of any external sensors.

When the robot approaches an obstacle its bumper micro switch triggers and the robot backs off for 2 seconds and then turns for half a second and the simulation resumes. This has been coded in all the robots on board processor itself. Similar logic also applies for boundary crossing avoidance. Its discussed later in the report.

PHASE 3

ITEM FORAGING

Item Foraging implies that the rovers are given commands in order for them to forge / find the items that are spread out in a given area. This part is directly linked with the algorithms which are run when the simulation in matlab is run. The L Alliance and Q Learning algorithms are run and depending on the algorithms commands are sent to the rovers based on virtually whatever happens in the simulation. The way a robot comes to know it is near an item is that the camera updates the notepad file with coordinates of the rovers and the item zones coordinates are found out in the initial part of the experiment. Then it calculates the distance between each of the items and the rovers. The ones which are closer to the item zone within a fixed amount of distance are sent for another check as per the algorithm. Finally we have the robots which are eligible to pick up an item from the item zone.

GRABBING MECHANISM

When a particular robot approaches an item zone, the robot waits and asks the camera about its orientation. It aligns itself such that it is in line with centre of the item zone. The item to be picked up here is a small piece of magnet which is kept in the centre of the item zone. The robot has a small strip of steel attached underneath its front side which is used to attract magnets. The robot is then sent a command to move forward a specified amount then move backward the same amount. During this time the entire simulation is paused. After the robot returns to its original position the simulation resumes. The item zone is virtually removed not physically. This is achieved by deleting the entry from a particular matrix in Matlab. The item zone is a red / black coloured square piece of paper in the centre of which is placed the magnetic item to be picked up.

After picking up a specific item the robot has to go to the target zone and drop the item. The dropping task can be indicated by glowing of an LED when the rover reaches the target zone. Another task to be handled is the physical cooperation between 2 weak robots to pick up a heavy item.

PHYSICAL COOPERATION

Cooperation is the process of group of robots (here) working together to achieve a common goal as opposed to working individually for their own goals. Physical Cooperation here implies that 2 robots who are physically weak can accomplish the task of picking up a heavy item from its zone. Physical Cooperation here can be achieved by the following ways –

- one robot picks up the item and the other robot follows it till the item is deposited in the target zone
- one robot picks up an item while the other robot stays near the item zone and later when item is deposited in the target zone they both are free to go
- one robot picks up an item and both the robots reach the target zone together by following any path and then after deposition of the item both of the rovers are free

OBSTACLE AVOIDANCE AND BOUNDARY DETECTION

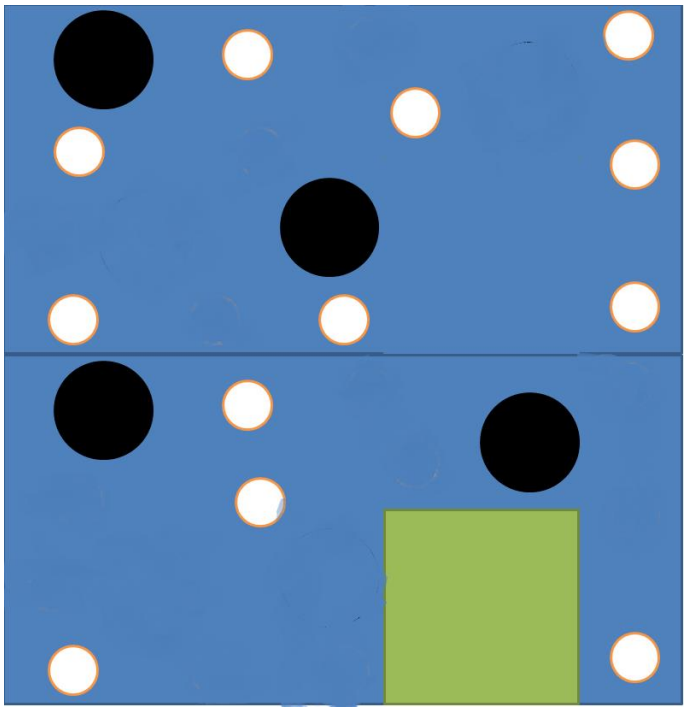
Initially it was thought that obstacle mapping and boundary detection would also be done using the overhead cameras. But as the final goal of the experiment is to minimize the use of external sensors as much as possible, the obstacles were avoided using the localized bumpers attached to the micro switches. Whenever a front bumper (micro switch) is triggered the robots are instructed to move back for 2 seconds and turn around for another 1 second. The same mechanism is being used for preventing the rover to cross boundary also. The entire area is being planned to be bounded by a thick rope / steel plate. So when the rover hits the boundary of the area, the rover would do the same action of backing off and turning around.

FINAL MAP

Finally the dimensions of the items zones, target zone and available area had to be altered from the targeted values. Expected area size for performing the experiment was 10 meter by 10 meter. But due to the lack of such a given space practically in the lab this area was reduced to 7 meter by 7 meter. Actually if such a space is practically available, by making slight modifications in the code a 10 by 10 meter area is also achievable with the help of 6 cameras. One camera covers 7 *3.6 metres so 3 cameras when placed side by side would be able to cover 10.8 meters and 3 of them when placed at the bottom can cover 14 meters in the vertical direction, but we can make a part of them to overlap and achieve the targeted area of 10 * 10 meters. Similarly by using 8 cameras side by side and on top of each other (4 + 4 on top of each other) we can also achieve an area of 14 by 14 meters.

The targeted dimensions for the items were 50 centimetres in diameter, but due to decrease in area from 10 by 10 to 7 by 7 meters; the item zone was reduced to 40 by 40 centimetres square. Similarly the target zone was reduced in size from 2 by 2 metres to 1.5

by 1.5 metres. The obstacle size remained unchanged at 1 meter by 1 meter. Finally the map looked as shown below:



FUTURE APPROACHES TO THE SAME PROBLEM

The same experiment can also be performed in complete darkness using different coloured florescent markers on top of the rovers and for item zones also. This would make the entire experiment light independent. And the results would be more accurate as compared to when using light and coloured papers.

As the ultimate aim of the project is to totally get rid of external sensors, it would useful to use the on board camera as well. The only external sensors being used in the experiment are the overhead cameras. The knowledge that the overhead cameras give is the localization of robots and mapping of items and target zone. The localization of the robots can be done by stepper motors in the future which may prove to be accurate. The mapping of the items may not be required as the robot can find an item based on triggering of the micro switch and then it takes a picture using the on board camera to know about the type of the item. An alternate way could be to use 2-3 IR sensors on the robots vertically mounted and know about the item to be picked up using these IR sensors. The item would be of 2 different heights. This is just a possible way which could be followed, although it requires adding additional IR sensors on the robot but it gets rid of the on board camera.

****