MULTIPLE ROBOT SYSTEMS:
Task Distribution, Coordination and Localization

Sameer Singh
83 ECE 2000
Final Year, NSIT

March 2004

Contents

Introduction to Multiple Robot Systems

The use of robots in day to day life is increasing faster than ever before. From applications in the manufacturing factories to education, from simple household chores to agricultural applications, from gathering information in mars to toys for the amusement of the wealthy; the list just goes on. Robotics, as an educational field, presents an interesting mixture of various fields, including electrical engineering, computer science, mechanical engineering, biology etc. Research into the field of robotics is also prevalent in every engineering research institute worthy of its name all over the world.

But robotics in general faces some major drawbacks. Firstly, all robots have to be very application specific, that is, a robot useful for one purpose cannot be used easily for some other purpose. This leads to an obvious loss in the finances. Secondly, most tasks expected from the robots require a very wide array of sensors and other resources, resulting in each robotic unit costing a lot. Lastly, all the robotic systems are not fault tolerant, that is, a faulty sensor or connection calls for a long delay and costly repairing. For all these reasons the development of robotic systems, and their application in real life, has been slower than expected.

Multiple Robot Systems address all these issues, and provide further benefits than the conventional single-robot systems. Before both the systems are compared, a brief description of a typical MRS (Multiple Robot System) shall be explained. As its core, it consists of a group of basic robotic units currently used for simpler purposes (like simple mobile units, and basic robotic arms). The number of the robots in the group may or may not be fixed. These robots should also be capable of communicating, with enough bandwidth as required by the MRS design. There are other optional components which vary from objective to objective, like the sensors required on each robot, sensors for environmental variables, central monitoring processor, etc.

At first, the MRS does not seem to economize any further from the traditional single-robot system; on the other hand, it seems to increase the price of the total setup. But when compared to the advantages it provides over the single-robot system, it is a small price to pay (literally). By distributing the computation and manipulation tasks over a number of robots, the speed of task completion is greatly increased. MRS are also more resistant to failures in the robots, for the simple reason that if a faulty robot is present, the other robots of the group shall replace it, slowing down the task execution, but not stopping it. The original speed shall be restored once the faulty robot is fixed and accommodated into the group. The robots used in MRS are very general purpose robots, and thus can be used for different purposes, saving cost of new robots this way. Further more, there are some tasks known as Tightly Coupled Tasks, which require high coordination between the robots, and are impossible to carry out by a single robot.

If the advantages provided by MRS are still not obvious, a proof is provided by nature. A huge amount of teamwork is prevalent in all species which carry out specific tasks. The foremost example in this respect is ants (an example which shall be referred to throughout the document). Individually, ants are much smaller and weaker than individuals of other species. They are not capable of communication of any sort, other than through pheromones, i.e. smell. They have very restricted vision, but a strong sense of smell. They have a small brain, which is incapable of anything more than a few actions. Analogically, ant can be thought of as a simple mobile robot, with a simple carry gripper, a sensor corresponding to the smell sensing, a very crude basic controller and a broadcast transmitter and receiver.

These very ants, starting from finding and carrying, end up collecting huge amounts of food resources at their base. Also, as is evident, the ant groups are very tolerant to loss of ants. Even if large bunches of ants (20-30) are removed from a trail of ants, the system slows down for about 10 seconds before resuming work, as if nothing had happened. Thus if a similar system is emulated using the above mentioned robot, the resulting system shall give all the above mentioned advantages. If this multi-agent system was not feasible and economically viable, ants would not be the one of the largest species on the planet, and still using almost negligible amount of resources.

This document describes the Multiple Robot Systems, in reference to the Task Distribution, Coordination and Communication required in these systems and examples & applications of MRS. The document refers to few of the latest developments in this field in research labs, and is an attempt to explain MRS. Thus the mentioned methods may not the best possible solutions.

Section Summaries

Given below is a brief description of each of the topics. These descriptions are only for reference and do not provide explanations or detailed information.

Mobile Robots for MRS
In this section, an industrially manufactured mobile robot which can be used for Multiple Robot Systems is introduced and described, namely the Pioneer 3DX.

Task Distribution
An *auction* based method of task distribution is described, which results in an efficient distribution of tasks among the various robots. Experimental results are also described, which include both the loosely and tightly coupled tasks.

Spatial Interference
In a densely populated environment, robots are likely to bump into each other. A conflict resolution method is given which offers several benefits like no communication overhead, simple computation, anonymity etc.

Localization
Absolute and relative localization is compared, and an egocentric relative localization method is studied. Experiment results are shown to demonstrate how positions of other robots are predicted based on the communication and the update rules.

Mobile Robot for MRS

Though Multi-Robot Systems does not necessarily mean using a mobile unit, the application of MRS is most prominent in area coverage, mapping, resource transfer, and the likes, which all involve mobile robots as their core. The requirements of all these applications can be met with the mobile robot which shall be discussed next, showing how single robot can be used for different applications unlike in single robot systems.

This robot was chosen to be described not because of any personal preference, or company sponsorship (unfortunately), or because it is the best. The Pioneer 3DX has been described because it is one of the most popular mobile robots, and one of the most used robots in multi-robot research.

The Pioneer 3DX is an agile, versatile intelligent mobile robot system, which includes many features internally, along with a wide of options available as extra accessories. The robot comes with a ring of 8 forward sonar. It stores up to 250-Watt hours of hot swappable batteries. It can carry a payload up to 23 kg, and reaches speeds up to 1.6 m/s.

Fig – The Pioneer 3DX

The bare P3-DX robot is capable of the following :

1. WANDER randomly
2. DRIVE, controlled by keys or joystick
3. PLAN PATHS, with gradient navigation
4. DISPLAY a map of its sonar readings
5. COMMUNICATE SENSOR AND CONTROL, i.e. information relating to sonar, motor encoders, motor controls, user I/O and battery charge data.
6. RUN C/C++ PROGRAMS, without using external software
7. SIMULATION SOFTWARE for testing on the desktop

The robot can be used for a wide range of functions due to the accessories available to it, like:

1. Wireless Ethernet
2. Laser Mapping & Navigation
3. Robotic Arm
4. Pan-tilt-zoom color camera
5. Stereo range-finding camera
6. Color-Tracking
7. Compasses & Tilt Positioning Sensors

8. Range finding Infra-red sensor
9. Internet based operation
10. Docking Station
11. Bumpers and Grippers

Other than these accessories there is provision to add up to 3 PC104+ cards. The robot has all these features packed in a small 44cmX38cmX22cm aluminum body.

Due to all these reasons, Pioneer 3DX is an ideal choice of usage in multiple robot systems. Only using a few changes in configuration, it can be used for a variety of applications, like, mapping, tele-operating, localization, monitoring, reconnaissance, vision, manipulation, etc. Since the robot provides ready to use hardware and a support for C/C++, the researchers have to concentrate only on the programming aspects of the multiple robot systems.

There are a few more robots of these kinds currently available for use. Robots like Khepera II and K-Alice have been for the purpose of education only, and do not offer high mechanical properties when compared to the Pioneer-3DX. They are cheaper and easier to use with compared to the P3DX, and can also be easily taken to demonstrations etc. The makers on Khepera and K-Alice (the K-Team) have also released Koala, which uses the same controller as Khepera, but can be used in realistic environments, harsher than what P3-DX can stand.

Fig – Khepera II robot

Fig – Koala Robot

These robots are currently used for single robot applications. Once multiple robot systems start being applied more in the industry, streamlined versions of these robots will be produced for MRS.

Task Distribution

The most important aspect of any Multiple Robot System is coordination, i.e. how the tasks can be distributed to different robots. Without efficient cooperation MRS is useless and a huge waste of resources. Efficient coordination should mean robots having different resources (like sensors, communication units, etc.) should be carrying out the tasks that fully utilize their features. Also, this coordination has to take place in a dynamic, noisy environment with fault prone robots.

Thus the problem now is that of task distribution and allocation. Once the tasks have been allocated to individual robots, the controller of the robot is equipped to carry it out. A method to carry out this task allocation shall be described next, followed by a simple test scenario. The author predicts a world where a large number of robots, most with different resources with them, have to carry out different tasks from time to time. The method is compatible to such a scenario. It attempts to minimize resource usage, task completion time, and communication overhead. Before further discussion, it should be noted that this problem is an Non-deterministic polynomial complete problem.

The method relies on *task commitment*, based on a *publish/subscribe* broadcast communication. For task allocation, it uses an auction based task declaration; Bidding takes place, and the task is allocated to the bidder with the highest fitness level. This solution to the problem of task allocation may not result in the ideal solution, but is a highly optimal solution, considering the fact that it is applicable to a huge number of heterogeneous robots.

The assumptions of the method are:

- the system is composed of physically embodied robots
- the robots are heterogeneous, possessing different skills
- the robots can communicate, but messages may be lost
- the robots are honest and cooperative
- the robots (or parts of them) can fail at any time
- a robot may not be aware of its own (partial) failure
- the robots are multipurpose, not task specific
- no model is available to describe the sequence in which tasks are generated
- if a robot is to oversee a task, it can determine its own progress and completion of the task.

The communication model for the method is broadcast. This model scores over directed communication (in which the robots have different identifications) in two ways. Firstly, unicast communication is very inefficient in delivering messages to multiple recipients, i.e. it is heavy on the controller and hogs a huge amount of bandwidth. Secondly, the robots in a multiple robot environment may move in and out of the communication range, or may develop faults which results in them not receiving and transmitting directed messages. Also, the number of robots in a

scenario is also not fixed. Thus anonymous unicast communication is preferred for MRS.

The tasks have been represented as a tree, i.e. each task is actually a tree containing other tasks. The parent task is broken into simpler child tasks, and the robot having this task has the responsibility to allocate them, and subsequently monitor their progress. Thus complex tasks can be easily represented containing simpler tasks as children, where simpler tasks are also trees containing much simpler tasks as children, and so on. Thus every task allocated to a robot contains further tasks to be allocated by the robot, except for the basic simplest tasks.

The allocation of the tasks follows an auction-like model. Auction method was chosen because auctions are scalable, allows for compartmentalization of data and control, and are efficient in computation and communication. The message to start an auction consists of a subject and content. All the robots "interested" in the subject may go through the content, and there after, bid.

The subject may contain physical devices (e.g. *camera*, *gripper*, *range sensor*, etc.), or higher-level capabilities (e.g. *mobile*, *open doors*, *carry heavy loads*, etc.), or indications of current state (e.g. *idle*, *out of charge*, *pushing box*, etc.). Thus if we require a robot which has to go to a point and observe something, we can send a message with the subject as {*mobile camera idle*}.

The initial allocation of a task in done by a human or a computer, but then is followed by task allocations by the robots. The task auction takes place in the following five steps:

1. Task Announcement – A message is sent with *announcement* as the subject, followed by details of the task, like the name, time, etc. It also contains another subject to aid further negotiation, i.e. list of resources.
2. Metric Evaluation – The announcement message, along with the list of resources, also contains a metric list, i.e. a list of metrics with which any robot can evaluate their fitness for the task. Metrics are chosen in such a way that they correctly evaluate a robot, and also have a very small probability of two robots having the same metric.
3. Bid Submission – Each robot evaluates themselves according to the metrics and transmits its *score* as a *bid* message.
4. Close of Auction – After a specific time, the auctioneer evaluates the bids, determines the winner, and notifies the robots with a *close* message. The bidder is awarded a time-limited contract to carry out the task, and the losers wait for more auctions.
5. Progress Monitoring – As the winner executes the task, the auctioneer constantly monitors the progress, and if satisfied, sends a *renewal* message (and receives an acknowledgement) till the task is completed. In case the progress is not satisfactory, the auctioneer re-auctions the task.

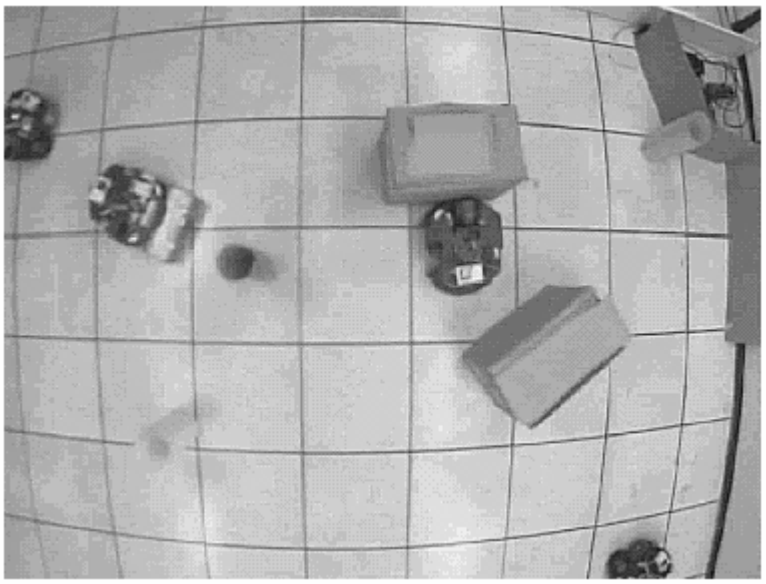


Fig – Different robot performing the different loosely coupled tasks

The method has been tested on two separate kinds of scenarios. Firstly a loosely coupled task environment was created, i.e. tasks that do not require a high amount of coordination, and thus, are basically single-robot in nature. A large room was used containing five heterogeneous robots, i.e. with different features attached to them. Without going into the details, the application of the method resulted in each of the robots doing what they were best at.

Next a tightly coupled task scenario was studied. The task required was to move a box from one position to another position. To carry out this task, the initial task was broken into two tasks, one of watching and supervising (*watchers*), and the other to do the actual pushing (*pushers*). This is analogous to the way humans move a huge box, i.e. when two people are carrying a box, they usually cannot see the destination, and thus a third person is required to constantly monitor and give instructions. Thus the higher-level task of watching contains child tasks of pushing the box. There are three robots in the setup, from which only one has a laser range finder, while the other two do not.

The initial task (of watching) is given to a robot capable of accurately perceiving (using a laser range finder) the angular error of box's orientation with respect to the path to the goal. Sending a message with *ranger* and *mobile* in the subject list does this. This robot then allocates the task of pushing to the box to the other two robots, and tries to make sure the angular error is zero by sending the *push-left* or *push-right* as auctions to the *pushers*. If the angle is already zero, then it just renews the contracts, i.e. they pushers should push as they have been.

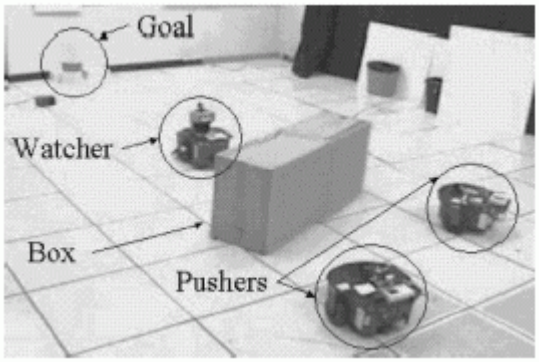


Fig – Tightly Coupled Task Scenario

The experiment was carried out with in different situations, and success/failure time was noted. The experiments included injecting and removing various complete or partial faults in the robots as they are carrying the box. The method showed that if an individual robot does fail, the non-faulty robot took up his place executed the task. This process can be understood more easily by observing the diagram of the box as it is pushed across the floor.

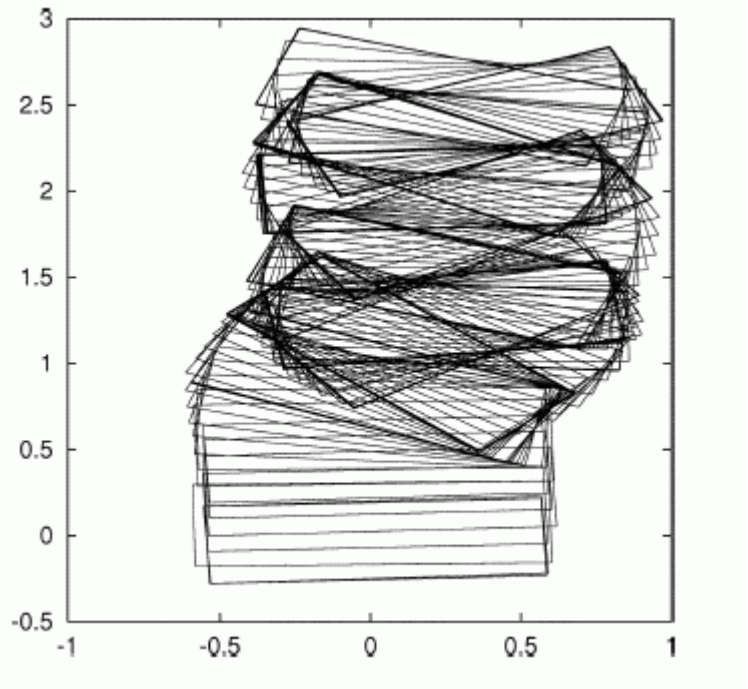


Fig – Path of box across the floor

Thus we see that this method effectively works in allocating the task to different robots in both loosely coupled and tightly coupled task scenarios, and also copes up with the different faults taking place in individual robots by auctioning their tasks automatically to more fit robots. This, or similar, sort of task distribution and allocation system is an inherent part of any Multiple Robot System.

Localization

The problem of localization is very important for any mobile multiple robot system, in fact it is very important for any mobile robot performing any sort of complex task. But localization techniques for multiple robots highly differ from the localization techniques used for single mobile robots.

Single robot systems use *Absolute* localization technique, that is, the robot calculates all coordinates relative to a fixed point (known as the origin). Absolute localization usually employs a GPS system or some other model-based method. This method requires either a high resolution GPS system or a very accurate and error-free odometry or inertial measurement kits to approximate changes in its own position. Errors in the sensing data has been tried to be corrected by applying Markov models and stochastic models of probability to calculate the positions. Experiments have been made to reduce error by placing beacons also, for the robot to correct its errors in coordinates by using different beacons to calculate it.

Absolute localization, though, is not very useful for multiple robot systems. Firstly in applications requiring mapping of unknown areas, using GPS system or other model-based techniques may not be possible (for e.g. mapping of Mars cannot include first placing beacons). Secondly, the absolute coordinates is not required for the robots to carry out a task, only relative positions are required. Using *relative* localization, i.e. when coordinates are calculated relative to one another, does not require high precision odometry instruments or expensive GPS systems. It is also easier on the processor, and does not require heavy computations.

Thus relative localization is the more preferred method for multiple robot systems. But before eliminating absolute localization altogether, it should be noted that relative localization can be calculated from absolute localizations, but the vice-versa is not possible.

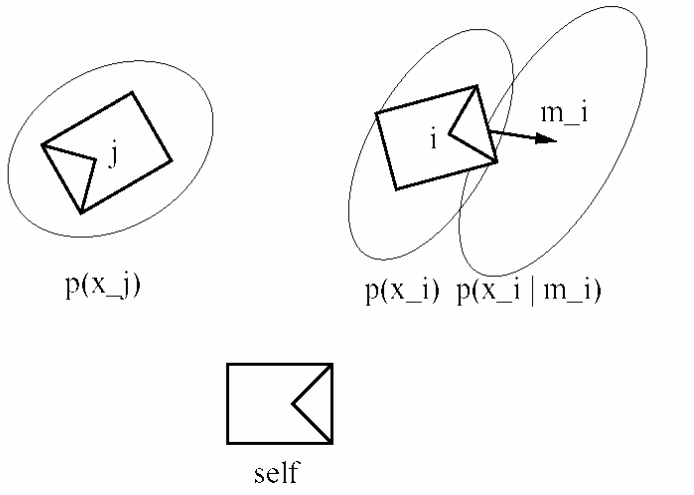
The relative localization technique which shall be described next is a complex method which takes into account many factors like the velocities, odometric faults etc. It is called the ego-centric approach, that is all coordinates are calculated relative to “itself”. The method does not require use of external landmarks or environmental models. It is highly decentralized, and yet requires minimum communication. It is able to easily self-initialise and self-correct. The only drawback is that it contains Markov models as its core, and therefore is memory and processor intensive.

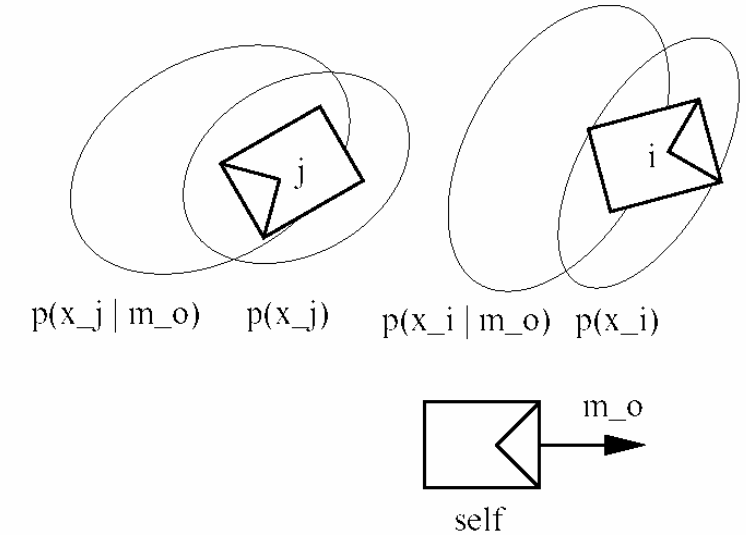
The method assumes the robots to contain robot sensors, which can measure the relative pose and identity of nearby robots, and motion sensors, to measure changes in its own pose. It also assumes efficient broadcast communication system.

Suppose there are *n* number of robots in a multiple robot scenario. Each robot then contains *n-1* distributions, corresponding to the positions of all the other robots. Let us consider one robot from the team, and refer to it as *self*. Also, let x_i denote the current pose of some robot *i* relative to self. The method shall now attempt to calculate $p(x_i)$ based on observations made by self and by other robots.

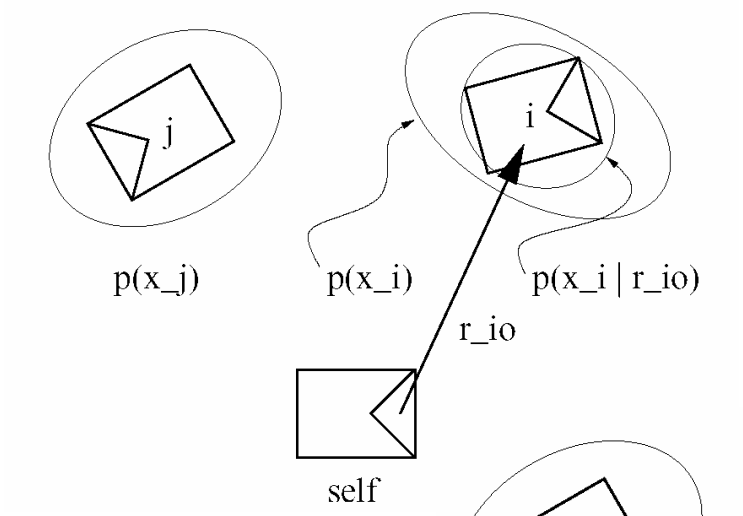
The sensors and communication system results in *self* generating five types of observations, using which the distribution of each robot is updated. The types of observation and the corresponding updating rules shall now be listed.

1. *Robot Pose Change* – If another robot changes its position, then the data is only relevant to the corresponding distribution. It shall be translated and then blurred (to account for uncertainty in observation).

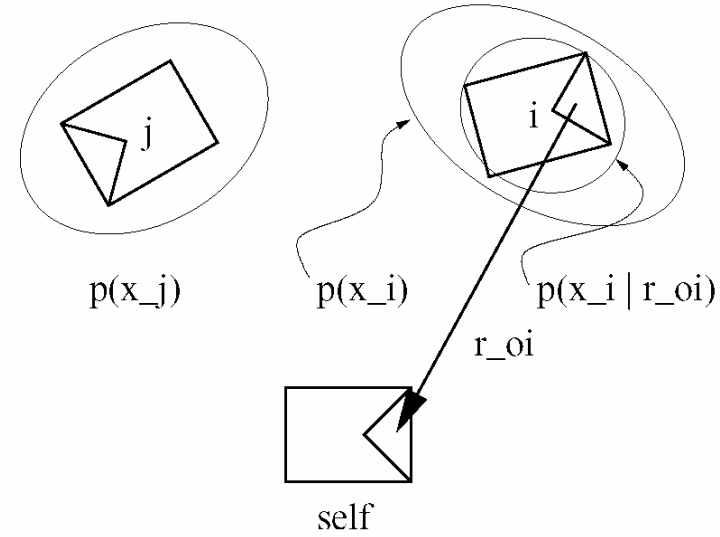




2. *Self Pose Change* – In this case all distributions shall have to be translated in the opposite direction, and blurred to account for the uncertainty in observation.



3. *Robot Observation by Self* – This observation shall only be applied to the probabilistic distribution of the robot being observed, by using the Bayes' Law, which generally sharpens the distribution.



4. *Self Observation by Robot* – This observation also shall only be applied to the distribution of the robot which has observed, and its distribution also shall sharpen. The sense of observation shall have to be inverted.

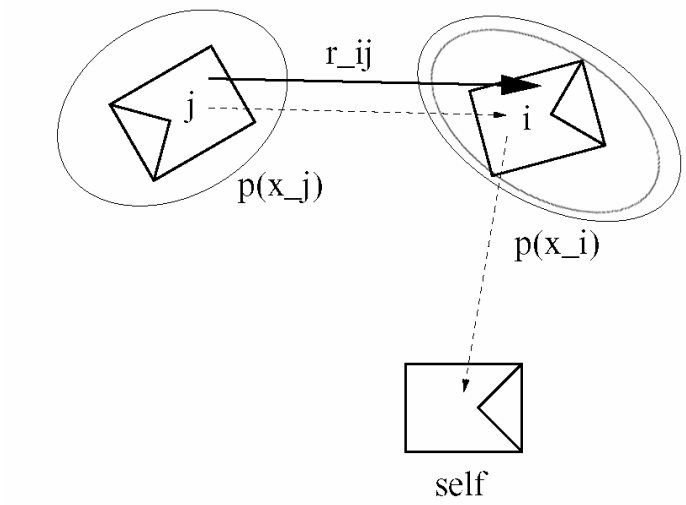


Fig – Dotted Arrows show the dependance

5. *Robot j observed by Robot i (Transitive)* – This case has to be handled carefully. Suppose first robot *j* is observed by robot *i*, which is used to update the position of *j*. Then robot *j* shall observe robot *i*, which shall then be used to update the position of robot *i*. This way, we have effectively counted the same observation twice, and used it to sharpen the distribution of both i and *j*, which may lead to inaccurate results. To avoid this circular reasoning, each robot contains a *n-1* node *dependency tree* in which all robots are represented as nodes. If a transitive observation is made, and if *i* is a parent of *j*, then $p(x_i)$ shall be updated. Otherwise if *j* is an ancestor of *i*, then $p(x_j)$ shall be updated. If they do not have a direct link, then the one with higher variance can be updated.

Using these update rules, after some time each robot shall have a converging distribution of each robot. Experiments were carried out by using four mobile robots equipped with the necessary sensors and communication system. The experiments resulted in the robots generating very accurate predictions of the positions of each robot.

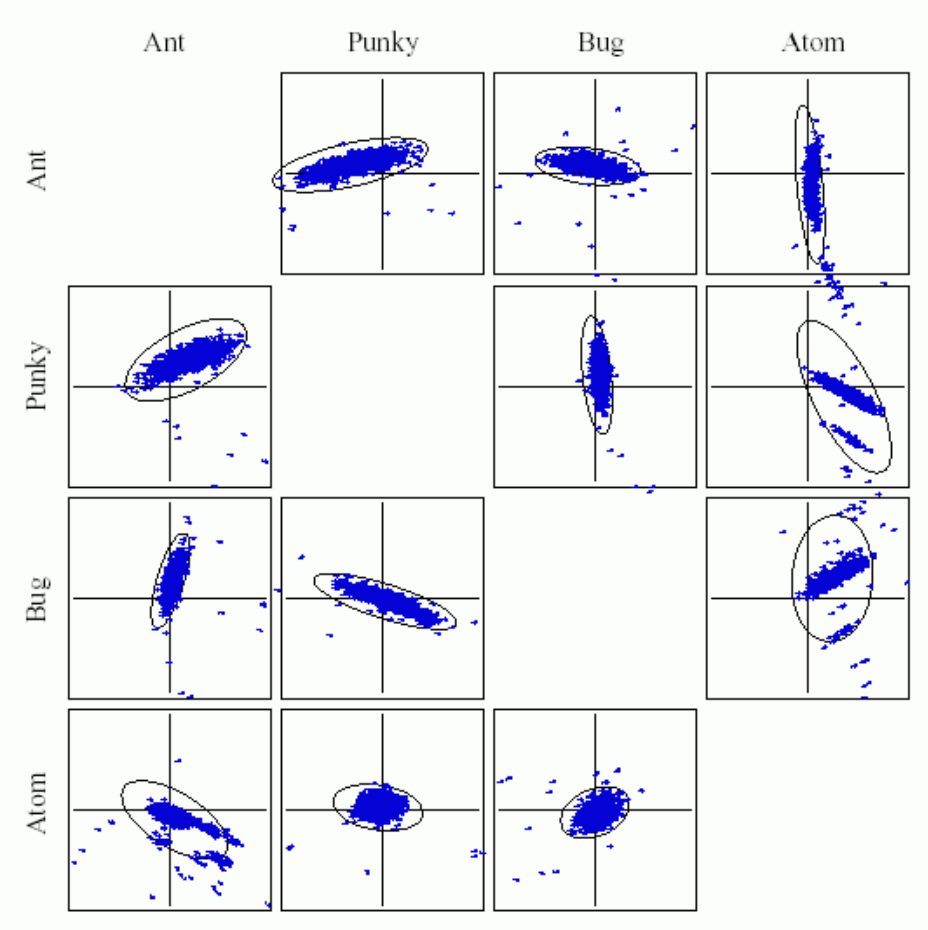


Fig – Ant, Punky, Bug and Atom are the four robots. This diagram shows the distributions of each of them of the others at a particular time. The centre of each cell is the actual position, and the ellipse is the 3ó interval for the distribution, i.e. the predicted position.

This method is highly useful since each robot knows the position of each other robot, even if it is not in sight, and can give commands as necessary. Though the experiment was of simple obstacle avoidance robots, this localization technique can be incorporated in any system to generate very useful data for the robots. Also, if one robot can calculate its GPS position, all robots shall know their GPS positions, even if they do not have the necessary feature. Thus this can be used even in very large multiple robot systems, which require GPS coordinates for all the robots.

This sort of relative localization can be used wherever mapping or area coverage is required, especially in USAR (urban search and rescue). It is also useful where a hierarchy is present in the robots, that is, the “leader” robot needs to know positions of all “worker” robots, so that it can issue commands accordingly.

Spatial Interference

Spatial Interference for single robot systems did not present much for a task since the obstructions were mostly stationary objects or objects with a predicted trajectory. But in the case of MRS, especially one envisaged earlier, containing a large number of heterogeneous robots all working on different tasks, the obstruction may be in form of other robots, with unpredictable paths. A different approach is required to approach this problem.

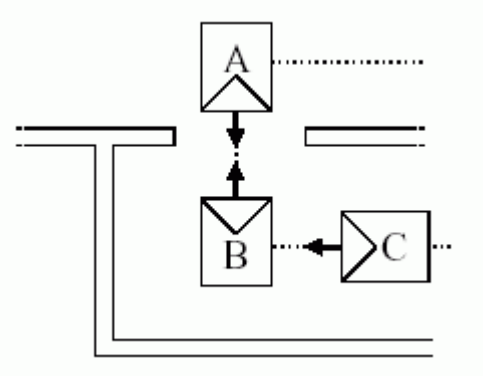


Fig – Locked robots situation

In a multiple robot scenario, especially in environments consisting of narrow corridor-like paths and doorways, it is very probable that a deadlock might take place, as in the figure. If both the robots just follow a avoidance rule, both may just oscillate inside and outside of each others' sensor area, without anyone letting the other one go. Thus this issue has to be addressed. This sort of problem usually takes place in resource transfer scenarios.

Anti-interference strategies, other than the one just mentioned, also fail. Bucket brigade strategy involves transferring the resource to be transferred to the other of the robot in the deadlock. This poses two serious problems. Firstly, robot to robot resource transfer must be incorporated, and there shall be a communication overhead. Also, this shall fail in an environment where different kinds of robots are working on different tasks. Another method which was suggested was to have a fixed dominance hierarchy tree in each robot, so that the one with the higher level can pass through. This method also is not feasible because it is not scale-free, and shall hog memory for a large robot system.

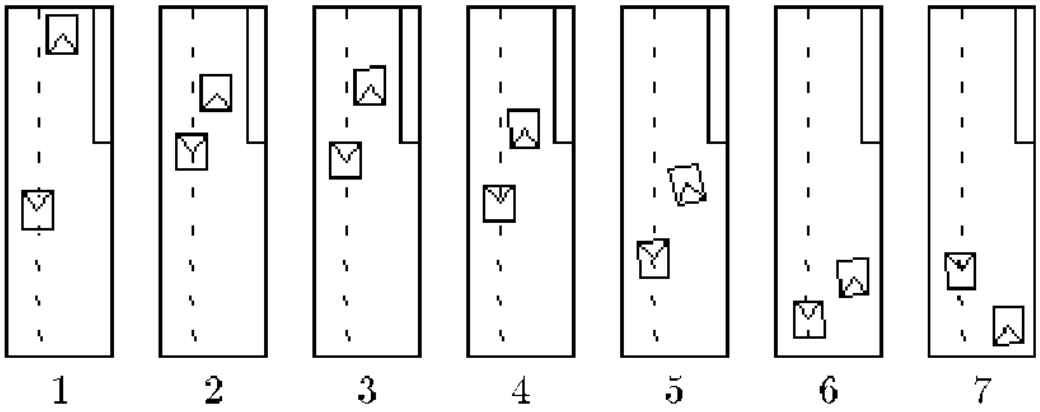
Thus a method is needed which is scale-free, decentralized, easy on the computations, does not have communication overhead, is independent of the navigation strategy and efficiently solves the interference problem. The method was discovered by observing nature, which in its own way is a large system of heterogeneous creatures, all working towards different objectives. If one creature meets another one coming from the opposite direction, there is usually a display of aggression. The more aggressive one then moves on, while the meeker has to let him. There is no fighting unless necessary and the problem has been solved.

This process of *Aggression Signaling* can also be applied to the robots to decide which robot shall pass, and which shall not. This method will work for a group of heterogeneous robots without any communication.

The controller of the robot shall contain three main behaviors.

1. *Navigate* – Path followed when no obstruction, i.e. it is executing the task.
2. *Panic* – To cope with situation that confound navigate, like non-robot obstructions, etc. Collision avoidance program, along with path recollection techniques may be stored here.
3. *Fight* – Is the conflict resolution method. Contains methods *brave* and *afraid* according to the *fear threshold*.

Navigate and Panic behaviors are self-explanatory. When the robot detects a robot within a fixed limit of the sensing range, it goes into the fight mode, and decides its *fear threshold* (between 0 and the fixed limit). If the other robot is in the fear threshold, the robot goes into *afraid* mode, and goes backwards as long as the fear threshold is invaded. Otherwise it continues its path in the *brave* mode for a fixed time before switching back to navigation after a fixed time limit.



The working can be understood by looking at the figure. 1) Two robots approach each other in a narrow corridor. 2) They get too close and make an emergency stop. 3) They go into *afraid* mode since they are too close, and back off. 4) The top robot reaches its fear threshold first and starts to move forward because it is not afraid anymore. 5-6) The lower robot is still afraid and keeps backing up till it reaches the end of the corridor towards the wider part. 7) The robots have passed each other.

Fear threshold for each robot can be decided using lots of factors. Some of them are listed below.

1. Random – Each conflict shall allow robots to generate their own random fear thresholds.
2. Personal Space – Total free space behind the robot.
3. Number of Previous Wins
4. Fixed Hierarchy of Robots
5. Priority of Task being Handled

A convenient solution can only be developed if a mixture of two or more of these factors shall be used to construct the fear threshold function. Work is still going on in this field to increase the efficiency of conflict resolution.

Concluding Notes

This document has been written as an attempt to introduce multiple robot systems. Various issues to be considered while designing and programming it were described, and examples of the technology used were given to further explain the problems faced. This section shall add to few of the minor issues, which cannot be left unmentioned.

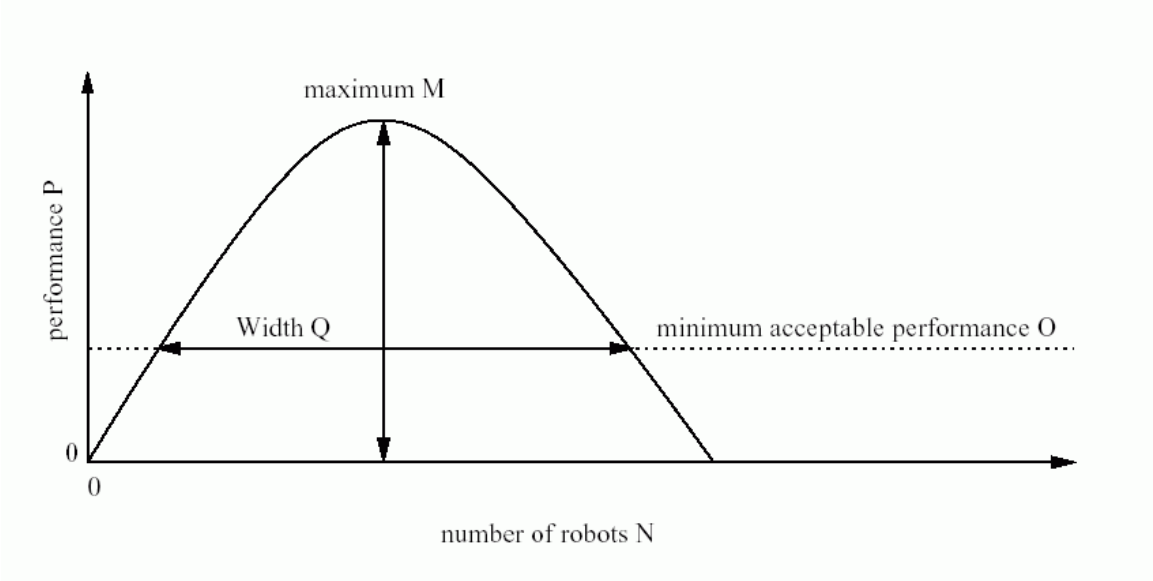


Fig – Hypothetical Graph of the performance v/s the number of robots in the system

The size of a Multiple Robot System is of great importance for the overall efficiency. By size, we mean the number of robots to be used in the system. The figure gives a hypothetical graph of how the number of robots and efficiency are related to each other. The curve touches the zero performance at least twice, once when N=0, and once when N = *Nmax*, e.g. when the whole area is covered with robots and there is no space for them to move.

The attempt should be made to reach the optimum population which provides the maximum performance. Some applications may have several peaks rather than a single peak shown here. Thus when any algorithm should be used in a multiple robot system, it should try and improve the performance curve by increasing M and/or Q, i.e. increasing the area under the curve above the minimum performance level. In applications where the population is dynamic, it should be tried to not let the number of robots drop below or increase more than the range of the number of robots required for minimum performance.

Another issue which requires thought is the *identities* of the robots, i.e. should the robots have personal identification identities, or should they remain anonymous, and non-identifiable. Each of the options provides various advantages and disadvantages. For e.g. while the anonymity causes problems in direct communications, there are problems of how the robots shall identify each other

(i.e. how they shall physically *convey* their ids to other robots) in the other method. This problem can only be solved by careful analysis of the objective, and thus is problem specific. In this document itself both the methods have been used in different places. A method could be created which gives temporary identifications to the robots coming in contact with the *self*, but the others remain anonymous. These ids should remain only for a specific period of time, and then erased from the memory. This is much like what happens in real life, when people are working with each other.

Type of communication method to be used is not a tough issue. Design should be made to minimize the required bandwidth, and a corresponding communication model is bound to be available, due to immense amount of progress made in this field.

References

- Vaughan, R.T., Stoy, K., Sukhatme, G.S. and Mataric, M.J. - *Go ahead, make my day: Robot Conflict Resolution by Aggressive Competition*
- Howard, A., Mataric, M.J. and Sukhatme, G.S. – *Cooperative Relative Localization for Mobile Robot Teams: An Egocentric Approach*
- Batalin, M.A. and Sukhatme, G.S. – *Spreading Out: A Local Approach to Multi-Robot Coverage*
- Gerkey, B.P. and Mataric, M.J. – *Sold!: Auction Methods for Multirobot Coordination*
- Mataric, M.J. – *Coordination and Learning in Multirobot Systems*
- ActiveMedia Robotics – *www.activrobots.com*

I would like to mention the USC Robotics Research Lab for the source of the inspiration and the source of these references. Most diagrams and figures I have used are from their publications.

Thank you

Sameer Singh
83 ECE 2000
Final Year, NSIT

March 2004