Behaviors of Multi-Robot Teams in MDLe

F. Zhang M. Goldgeier B. Madhusudan/P.S. Krishnaprasad

University of Maryland 1856

The Institute for Systems Research

Intelligent Servosystems Laboratory (ISL)

Multi-Robot Teams

- Three 2-wheeled mobile robots controlled by differential driving rules
- Communicating by using wireless internet connection
- Equipped with sensors including sonar, binaural head and camera

A test-bed for control, communication and sensing technique of use in Formation Control

Challenges for Existing Platform

Extensive work has been done on the development of a unified platform for implementing control algorithms on a single system. However, it is much more difficult to apply control algorithms to the formations of multiple robots. One may argue that a formation control algorithm can be modeled and implemented by running one copy of the existing platform on each robot. One problem with this is that one would have to implement the controller explicitly on each robot. Imagine the example of commanding a group of drilling soldiers. The commander will want to issue commands for the whole group instead of commanding each individual soldier. Another problem is related to the limited speed and bandwidth of communication links between the robots. We cannot afford to pass large amounts of state or sensor information across these links.

Motion Description Language(MDLe)

Motion Description Language (MDLe) provided a formal basis for robot programming using behaviors and at the same time permitted incorporation of kinematic and dynamic models of robots in the form of differential equations. Current MDLe platform has provided a framework for autonomous robot control and motion planning that separates the user from the low level implementation details of a specific robot. In MDLe, controlling command are represented by strings of **Atoms**, for example:

(Atom (bumper) (go 30 30))

defines an atom to drive a robot forward at speed 30 until it bumped into an obstacle. This will trigger the bumper sensorsto interrupt the atom. Here, ``bumper'' and ``go'' are **quarks** which can be shared by other atoms.

One Command for the Whole Team

In order to extend MDLe to be able to control multi-robot teams, our idea is to separate the group level maneuver commands from the group keeping commands. We also separate the events that require the whole group to take action from the events that only require a member to respond. A group level maneuver command will be translated into commands for each member of the team . We define a **group atom** to be the atom where only group level maneuvering command and group level interrupts(responses) are implemented. We define a **shadow atom** to be the atoms that each team member will finally execute. The mapping from group atoms to shadow atoms will allow formation keeping controllers to take effect. Mappings can be written a priori by designing proper quarks. For instance,

(GroupAtom (Bumper) (go 30 30))

is a group atom which will drive a group of robots forward until any of them hit an obstacle. In order to keep all the robots information, assume one of the robots is lagging behind its nominal position, the shadow atom on this robot might be

(Atom (Bumper) (go 31 31))

where the number ``31'' is obtained by a mapping quark where formation control is implemented. Depending on the formation strategy employed, the mappings can be identical or different for different robots. In a leader-follower case, the leader will run the group atom directly but the followers will map the group atoms into proper shadow atoms depending on where the follower is.In the behavioral or structural approach the mappings will be the same on all the robots.

Team Response to Outer Events

An interrupt is the point at which switching occurs from one control law to another. Usually, an interrupt takes place as a result of obtaining certain readings from sensor measurements. In order to implement a group level interrupt, one would need to know the sensor readings from all of the robots involved. As more robots are added to the group and more sensors are used on board each robot, the information required to determine group interrupts increases in size. Furthermore, since the interrupts need to be evaluated at high frequency in MDLe, the sensor communications would need to be performed over a very short period of time.This problem can be resolved by designing an inter-robot communication language where different combinations of sensor readings can be represented as different words. For example, instead of sending all of its sonar readings to the group for each individual robot to process at every turn, a robot can broadcast the word "obstacle" only when an obstacle exists in front of itself by processing its own sensor data. Thus, the information is not sent at every turn and processing time is saved.

Language for Coordination

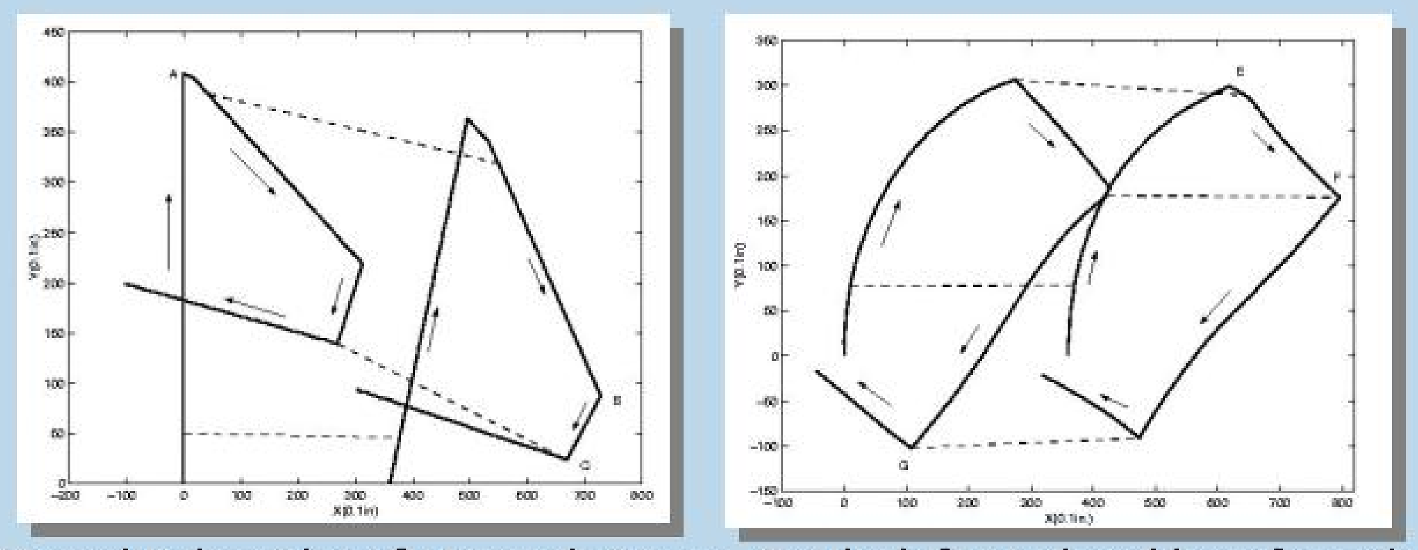
We have designed a message passing and processing mechanism. Messages are arranged as various field/value pairs. Each message must begin with a **msgstart** field and end with a **msgend**. Each message sent consists of a required header followed by zero or more name/value pairs of data. The header is made up of the following fields: TO - The name of the robot we are directing the message to. Messages directed towards the entire group are sent to the name "<broadcast>". FROM - The name of the sending robot. SUBJECT - The message's subject. This subject determines how the message will be processed when it is received by other robots. It specifies the type of data that the message contains. LEVEL_ID - An integer identifying the plan level that the message is being sent to. This ID allows a hierarchical priority system that allows messages to be sent to atoms and plans at certain level on the receiving robot. A message being broadcast from a robot named "scotty" to the entire group in order to indicate that it has detected an obstacle in its path may look like this:

msgstart: ; **from**: scotty; **to**: <broadcast>; **subject**: obstacle; **level_id**: 0; **nominal-angle**: 30; **msgend**:

Faster Communications

The mapping from group atoms to shadow atoms requires the knowledge of the state vectors from other robots. This can be obtained by reading from the communication links. Since most of the formation controllers require real-time feedback of state vectors, the communication will happen at very high frequency. Because a time slice is very short, the communication link must be sufficiently fast to ensure all necessary information be exchanged. A specialized network layer protocol is under development

Experimental Results of 2Robot Team



Ground trajectories of a two-robot team. On the left, no closed loop formation control is implemented. The robots failed to keep the formation after obstacles are detected at position A, B, and C. On the right, group atoms and shadow atoms are used to incorporate the formation maintenance controller. The formation is successfully maintained