Velocity Consensus of Multiple Robots

Bagadi Tarun Kumar
1501013

Electronics and Communication Engineering Department

Indian Institute of Information Technology Guwahati

A Report submitted in partial fulfillment of
Bachelor of Technology

12^{th} November, 2018

Certificate of Approval

This is to certify that the 7th Sem B.Tech. project report entitled **"Velocity Consensus Multiple Robots"** is a record of bonafide work carried out by **"Mr.Bagadi Tarun Kumar"** under my supervision and guidance.

The report has partially fulfilled the requirements towards the degree of Bachelor of Technology in Electronics and Communication Engineering at Indian Institute of Information Technology, Guwahati.

Signed:

Dr. SURAJIT PANJA
Assistant Professor
Department of Electronics and Communication
Indian Institute of Information Technology, Guwahati
Guwahati 781001, Assam, India.

Contents

Chapter 1

Introduction

Consensus is of the one of the most fundamental collective behaviors of a group of autonomous dynamical systems. Each dynamical system is known as an agent and the collection of all such agents is known as multi-agent systems (MAS). The basic idea for consensus of MAS is that each constituent agent updates its control input based on the information states of its local neighbors in such a way that the nal states of all agents converge to a common value of certain physical quantities. In other words, the control task on reaching this agreement on the value of the quantity can only be realized by cooperative distributed control laws that takes into account only relative information, i.e., state information of the agents with whom the agent under consideration shares direct communication. This type of control is also known as cooperative control as the agents cooperate or collaborate with one-another to realize the control task at hand.

There are two types of consensus that are mainly studied in the existing literature namely leader-follower consensus and leaderless consensus. In leader-follower consensus, the leader, unperturbed from external control, acts as the reference trajectory generator which the other followers in MAS follow. In most cases, the leader specifies a fixed value of convergence on which the followers must reach on. On the other hand, in a leaderless consensus, there is no specied reference agent. The advantage of the former is that the consensus convergence value can be specified and fixed beforehand.

The study of consensus for dynamical mobile robotic systems directly alludes to velocity and/or acceleration consensus. In the present study, the focus is on reaching velocity consensus of a group of mobile robots. To that end, agents are required to communicate to estimate or get the velocity of their neighboring agents.

The ultimate aim of velocity consensus is to attain the same velocity. Velocity consensus can be used for a convoy of vehicles of military and civilian use. This type of consensus under study can be achieved by a group of robots that follows a leader that specifies for the group what common velocity to achieve in order to avoid collision.

Agent

There is no strict definition about what an agent is. Literature offers a variety of definitions ranging from the simple to the lengthy and demanding ones. All available definitions are strongly biased by the background field interested in agent technology like artificial intelligence, software engineering, cognitive science, computer science, engineering in general, to name a few. One of the definitions of an agent runs as follows:

"An agent is anything that can be viewed as perceiving its own environment through sensors and act upon that environment through effectors".

According to this definition an agent is any entity, physical or virtual one, that senses its environment and act over it.

Physical entities that could be considered agents, say in the case of a power system, are simple protection relay or any controller that controls directly particular power system component or part of the system.

Virtual entity that can be considered an agent is a piece of software that receives inputs from an environment and produces outputs that initiate acting over it.

An agent is called autonomous if it operates completely autonomously, that is, if it decides itself how to relate its sensory data to motor commands in such a way that its goals are attended to successfully.

We require systems that can decide for themselves what they need to do in order to achieve the objectives that we delegate to them. Such computer systems are known as agents. Agents that must operate robustly in rapidly changing, unpredictable, or open environments, where there is a significant possibility that actions can fail, are known as intelligent agents, or autonomous agents.

An agent is defined as a computer system that is situated in an environment that is capable of autonomous actions in this environment to meet its design objectives The key point about agents is they are autonomous:capable independent action.Thus an agent is a computer system capable of autonomous action in some environment, in order to achieve its delegated goals.

We think of an agent as being in a close-coupled, continual interaction with its environment:

$$\text{sense} - \text{decide} - \text{act} - \text{sense} - \text{decide}\ldots$$

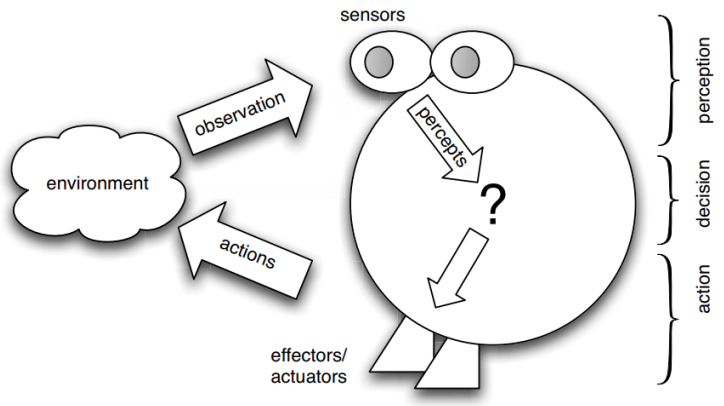


Figure 1: An agent in its environment. The agent takes sensory input in the form of precepts from the environment, and produces as output actions that affect it. The interaction is usually an ongoing, non-terminating one.

A Multi-agent system is one that consists of a number of agents, which interact with one-another. In the most general case, agents will be acting on behalf of users with different goals and motivations. To successfully interact, they will require the ability to cooperate, coordinate, and negotiate with each other, much as people do.

Advantages of Multi-Agent Systems:

Main advantages of MAS are robustness and scalability. Robustness refers to the ability that if control and responsibilities are sufficiently shared among agents within MAS, the system can tolerate failures of one or more agents.

Scalability of MAS originates from its modularity. It should be easier to add new agents to a MAS than to add new capabilities to a monolithic system.

The real world is a multi-agent environment: we cannot go around attempting to achieve goals without taking others into account. Some goals can only be achieved by interacting with others. Multi-agents bring about several advantages over Single Robot Systems. For example task of cleaning up hazardous waste, where the task allocation and coordination among the team involved. The agents interact with each other and coexist in an environment, that they perceive, resulting in a distributed world model. We are interested in fully cooperative multi-robot systems in which all robots have a common goal. Roles are a natural way of introducing domain prior knowledge to a multi-agent problem and provide a flexible solution to the problem of distributing the global task of a team among its members.

The role assignment not only reduces the number of actions that have to be considered for each agent, but can also be used to determine which agents depend Social ability in agents is the ability to interact with other agents (and possibly humans) via cooperation, coordination, and negotiation. At the very least, it means the ability to communicate.

Cooperation is working together as a team to achieve a shared goal. Often prompted either by the fact that no one agent can achieve the goal alone, or that cooperation will obtain a better result. Coordination is managing the interdependencies between activities. For example, if there is a non-sharable resource that you want to use and I want to use, then we need to coordinate. Negotiation is the ability to reach agreements on matters of common interest.

Chapter 2

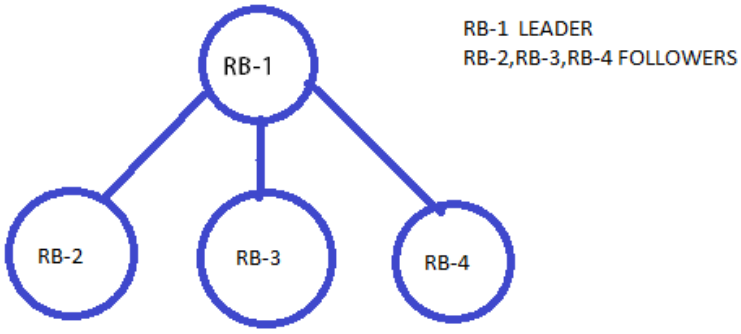
Motivation

Multi-agent systems are more desirable than single agent systems as they can perform a given task with greater efficiency. This is extremely crucial in many time sensitive applications. Moreover, they provide increased flexibility and scope of application. However, multi-agent systems need to be able to communicate with each other in order to complete a given task correctly. This communication must be extremely accurate, specially in applications like fault diagnosis, medical surgeries etc. Consensus of multi-agent systems is therefore an active area of research.

Chapter 3

Problem Definition

(i)The objective is to create an inter-communicating troop of robots such that all the other robots attain a common velocity upon receiving a signal from one particular robot designated as leader robot.



(ii)The objective is to create an inter-communicating troop of robots such that each of them can pass on a message to the next and eventually attain a common velocity by using a simple controller

Chapter 4

Work Done

1 QUICK VIEW OF ROBOT

Fire Bird VI platforms set the standard for intelligent mobile robots by containing all the components for sensing and navigation in real-world environment. This has become reference platforms in a wide variety of research projects. It comes in fully assembled and ready to use form. It has high precision DC gear motors with the high resolution position encoders. Motors are driven by smart motion control unit with velocity and acceleration control. By selecting correct locomotion drive, robot can be used in indoor and outdoor environment. Robot has ultrasonic range sensors with range of 6 meters, IR distance range sensors and line sensors and many more expandable interfaces. Robot is powered by high capacity battery pack. Fire Bird VI robot has two wheel differential drive and a caster wheel at the front side for the support. It is the most recommended configuration for accurate locomotion. This configuration is only used in the indoor environment. It has on board pc with external gps and imu, Servo Pod interfacing board.

HARDWARE SPECIFICATIONS:-

Weight of the robot 4.8 Kg

Power:-

System Power: 4 Cell, 16.8V Lithium Polymer battery

PC Power : 4 Cell, 16.8V Lithium Polymer battery

Sensors :-

8 x Ultrasonic sonar sensors with 6 meter range

8 x Infrared distance range sensors.

Motion control :- DC geared

Communication :- 1.USB serial communication.

2).wireless 2.4 GHz Wireless communication module(rf module)

Sensor module
Sensor module of Fire Bird VI robot acquires data from 8 line sensors, 8 Ultrasonic range sensors and 8 IR distance range sensors. It also generates PWM control signals for 3 servo motors. It can also turn individual sensor bank ON or OFF in order to conserve power or to allow multiple robots to work in the same area by synchronizing each others sensors.

Interference with other robot's sensors
If many robots are operating in the same environment then one robot's transmitted ultrasonic pulse may directly affect other robot's obstacle detection readings. In multirobot environment you can also coordinate the use of sensors with other robots using robot's on board 2.4 GHZ wireless module to insure that at any given instant only one is using its ultrasonic range sensors. For more information on commands to initiate ultrasonic sensor's to take readings, refer Software manual of robot.

Blind spot of the Ultrasonic Range Sensors
Ultrasonic sensors have range up to 6 meters. However it shows obstacles in the range of 0 inch to 6 inch as 6 inch. It can not detect obstacle closer than 6 inches reliably. To cover the blind spot, IR distance range sensor is used. In standard configuration robot comes with IR distance sensor which has sensing range of 10 to 80cm. It is used to cover blind spot of the ultrasonic range sensor

IR Distance Range Sensors
Fire Bird VI Version has been equipped with 8 IR distance range sensors. These sensors are used to detect the obstacle in the blind spot range of the ultrasonic range sensors. In standard configuration robot uses 10cm to 80cm IR range sensor. It has blind spot for the distance less than 10cm. On special request, robot can also be made available with 20cm to 150cm IR range sensors which has the blind spot of 20cm.

Interference with other robot's sensors
If many robots are operating in the same environment then one robots transmitter will directly affect other robot's receiver even from the distance of few meters. You can switch off power to the IR range sensor by sending specific command to the sensor module. You can also coordinate with other robots using robot's 2.4GHz wireless module to ensure that at any given instant only one robot's IR range sensors remain active

Motion controller supports following modes

Mode 0: Open loop velocity control This is the simplest mode of operation. In this mode, amount of power to be given to the motor can be specified. This is open loop control mode. In this mode robot has the quickest response but there is no velocity or position control.

Mode 1: Closed loop velocity control This is closed loop velocity control mode. In this mode, robot follows exactly the specified velocity. Robot will go exactly straight and will maintain its velocity even while climbing and going down the slope. It is the most useful mode in the robot.

Mode 2: Position control mode In this mode, robot goes from point A to Point B with specified distance. The distance is specified in terms of

encoder counts for each motor. Robot's maximum velocity and acceleration for the position control can be set.

Fire Bird VI Robot Communication

Fire Bird VI robot supports two different modes of communication, viz. USB (Serial) and 2.4GHz (wireless) module.

Serial communication USB serial communication is preferred for on board embedded PC or laptop. Connection between robot and on board PC/laptop can be established through USB cable. **Wireless Communication** For wireless communication, you should have wireless module inside the robot. Another wireless module will plug into USB port of computer from where you control the robot. Fire Bird VI robot has integrated 2.4 GHz wireless module with unique ID. Information related to wireless module settings are mentioned on top of Fire Bird VI robot near wireless module antenna.

Fire Bird VI Robot Communication Protocol

The robot communicates with host PC or embedded kit using serial or wireless interface. communication settings are as follows:

Baud Rate: 57600

Data Bits: 8 Parit

None Stop Bits: None

The communication between the host computer and Fire Bird VI robot is performed by using designated commands. The host computer acts as a master and Fire Bird VI robot acts as a slave. The communication is always initiated by host device. The host device should wait for receiving a response for sent command before sending next command. The communication protocol has a simple request and response structure. The Host PC requests a particular operation to which the target module responds according.

First three byte of each command are always "NEX" which are then followed by the command type and the appropriate command data bytes and checksum for error detection.

Command: Host to Fire Bird VI

Table 1: Fire Bird VI command

Header	Command	Sub-Command/Data	Checksum
N' 'E' 'X'	1byte	1byte	1byte

Fire Bird VI command packet In response to the command, the robot will start processing a task, and it will send a reply. The first byte of each reply is either "S" or "F" followed by the command type and corresponding response data as shown in the table below. Here the command type is specified in response to help host identify the response for particular command sent. The first byte of response indicates whether robot executed command successfully. Here 'S' represents Successful execution and 'F' represents failure in execution of given command.

Fire Bird VI. Response

Table 2: Fire Bird VI response

Header	Command	Data	Checksum
'S' or 'F'	1byte	nbyte	1byte

Programming and Support :- C/C++, Scilab, Python, Raspberry Pi
Operating system used- windows
Platform used - microsoft visual studio, FireBird VI GUI, serial terminal.
Programming Language used C
Operating system used- linux
Platform used – code blocks for c, python
Programming Language used python

2 METHOD OF COMMUNICATION

communication used is LCM- **Lightweight Communications and Marshalling (LCM)**

We describe the Lightweight Communications and Marshalling (LCM) library for message passing and data marshalling. The primary goal of LCM is to simplify the development of low-latency message passing systems, especially for real-time robotics research applications. Messages can be transmitted between different processes using LCM's publish/subscribe message-passing system. A platform and language-independent type specification language separates message description from implementation. Message specifications are automatically compiled into language-specific bindings, eliminating the need for users to implement marshalling code while guaranteeing run-time type safety. LCM is notable in providing a real-time deep traffic inspection tool that can decode and display message traffic with minimal user effort and no impact on overall system performance. This and other features emphasize LCM's focus on simplifying both the development and debugging of message passing systems. In this paper, we explain the design of LCM, evaluate its performance, and describe its application to a number of autonomous land, underwater, and aerial robots.
Supported platforms:- GNU/Linux; OS X; Windows; Any POSIX-1.2001 system (e.g., Cygwin, Solaris, BSD, etc.)
Supported languages:-o C C++; C; Java; Lua; MATLAB; Python;
We divide our description of LCM into several sections: type specification, marshalling,communications, and tools. Type specification refers to the method and syntax for defining compound data types, the sole means of data interchange between processes using LCM. Marshalling is the process of encoding and decoding such a message into binary data for the communications system which is responsible for actually transmitting it.

A. Type Specification:- Processes communicating with LCM agree in advance on the compound data types used to represent information. LCM does not support defining Remote Procedure Calls (RPC), and instead requires applications to communicate by exchanging individual messages. This restriction makes the LCM messaging protocol stateless, simplifying other aspects of the system (particularly logging and playback). To aid this process, LCM defines a formal type specification language for describing messages The binary representation of a message is implicitly defined by its type definition. These definitions are independent of platform and programming language, and a code generation tool is used to automatically generate language-specific bindings that provide representations of the message in a form native to the programming language.
B. Marshalling In order for two modules to successfully communicate, they must agree exactly on how to interpret the binary contents of a message. If the interpretations are different, the resulting system behavior is typically undefined, and usually unwanted. In some cases, these problems can be obvious and catastrophic: a disagreement in the signedness of a motor control message, for example, could cause the robot to suddenly jump to maximum reverse power when the value transitions from 07f to 080. In other cases, problems can be more subtle and difficult to diagnose. Additionally, as a system evolves, the messages may change as new information is required and obsolete information is removed. Thus, message interpretation must be synchronized across modules as messages are updated.
C. Communications The communications aspect of LCM can be summarized as a publish-subscribe messaging system that uses UDP multicast as its underlying transport layer. Under the publish-subscribe model, each message is transmitted on a named channel, and modules subscribe to the channels required to complete their designated tasks. It is typically the case (though not enforced by LCM) that all the messages on a particular channel are of a single pre-specified type.
1) UDP Multicast In typical publish-subscribe systems, a mediator process is used to maintain a central registry of all publishers, channels, and subscribers. Messages are then either routed through the mediator directly, or the mediator is used to broker point-to-point connections between a publisher and each of its subscribers. In both cases, the number of times a message is actually transmitted scales linearly with the number of subscribers. When a message has multiple subscribers, this overhead can become substantial. The approach taken by LCM, in contrast, is simply to broadcast all messages to all clients. A client discards those messages to which it is not subscribed. Communication networks such as Ethernet and the 802.11 wireless standards make this an efficient operation, where transmitted packets are received by all devices regardless of destination. LCM bases its communications directly on UDP multicast, which provides a standardized way to leverage this feature. Consequently, it does not require a centralized hub for either relaying messages or for "match making". A maximum LCM message size of 4 GB is achieved via a simple fragmentation and reassembly protocol. The multicast time-to-live parameter is used to control the scope of a network, and is most commonly set to 0 (all software modules hosted

on the same computational device) or 1 (modules spread across devices on the same physical network).

2) Delivery Semantics LCM provides a best-effort packet delivery mechanism and gives strong preference to the expedient delivery of recent messages, with the notion that the additional latency and complexity introduced by retransmission of lost packets does not justify delaying newly transmitted messages. In general, a system that has significant real-time constraints, such as a robot, may often prefer that a lost packet (e.g., a wheel encoder reading) simply be dropped rather than delay future messages. LCM reflects this in its default mode of operation; higher level semantics may still be implemented on top of the LCM message passing service.

PUBLISHER CODE

Import headers ;
Initialization of variables ;
Initialization of communication;
Define motor parameters ;
Open communication port ;
Set motor parameters mode ;
IF motor parameters and communication established;
Set left motor velocity ;
Set right motor velocity ;
Motion move forward ;
Get Left motor velocity, store in msg.t1;
Get Right motor velocity , and store in msg.t2;
Delay;
MSG.ENABLED =TRUE;
Publish message ("example ", msg.enable);

Subscriber code

Import header;
From exclm import example$_t$;
$Intilizationofcommunication$
$Definemotorparameters$
$Opencommunicationport$;
$Setmotorparameters$;
$Settopostionmodewithaanyvelocity$
$IFmotorparametersandcommunicationestablished$
$If(msg.t1andmsg.t2recevied)$
$Settheleftmotorvelocitytomsg.t1$
$Settherightmotorvelocitytomsg.t2$
$Delay$
$Lc = lcm.LCM()$;
$subscription = lc.subscribe(\text{ı}examplemyhandler\text{ȷ})$

Chapter 5

Conclusion and Future Work

Conclusion Velocity consensus has been achieved among the agents by making coommunication among them .

In recent years, commercially available mobile robots, that operate in indoor environments, have found their ways into private homes, office environments, and industrial settings. They fulfill important duties such as transportation,cleaning. While some tasks, such as vacuum cleaning, can be achieved with rudimentary perception, for more complex tasks sophisticated perception capabilities are fundamental for autonomous robots. Even for the mentioned vacuum cleaning, improved perception allows for substantial increases in efficiency. In this thesis, we investigate novel approaches for the modelling of indoor environments for robot navigation. Being an important foundation for higher level skills, a particular focus lies on simultaneous localization and mapping (SLAM), which allows a robot to construct a model of its environment during operation. In the context of SLAM, we develop an approach for RGB-D cameras/LIDAR that captures dense 3D maps for robot navigation. For this SLAM system, we propose novel methods to increase the accuracy of the trajectory estimation and the robustness against misassociations during individual motion estimates. Further, we address a major limitation on the hardware side of RGB-D cameras/LIDAR namely the limited field of view. We investigate SLAM with multiple RGB-D cameras /LIDAR and develop an approach for automated extrinsic calibration of RGB-D cameras/LIDAR via SLAM.

List of references

[1] Jiming Liu,Jianbing Wu, *"Multi-Agent Robotic System"*.CRC Press 2001

[2] Gehard Weiss, *"MultiAgent Systems"*,second edition,The MIT Press,1999.

[3] *Fire Bird VI Hard Ware Manual.*

[4] *Fire Bird VI Soft Ware Manual.*

[5] https://lcm-proj.github.io/