Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 48-29 (2015) 170–175

Redefining voting to enable simultaneous multioperator robot navigation

Lakshminarasimhan Srinivasan * **Pablo Baier** *
Christian Herrmann * **Julian Scharnagl** * **Klaus Schilling** *

* *Department of Robotics and Telematics, University of Wuerzburg, 97074 Wuerzburg, Bavaria, Germany.*
E-mail: (srinivasan, baier, herrmann, scharnagl, schi)@informatik.uni-wuerzburg.de.

Abstract: This paper details an algorithm where multiple users can simultaneously and efficiently operate on a single remote robot. Single user single robot remote operation is fairly advanced but multi-user operation is faced with many drawbacks, especially when users are spatially seperated and have no means of knowing what commands are being issued by other operators at the same time. This is a non-trivial problem and has so far always been solved using human intervention using built in voting mechanisms. This paper is focussed on using an intelligent voting algorithm which dynamically computes a vote based on operator commands without the need for any manual intervention. Extended results are documented and analysed.

© 2015, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.

Keywords: Tele-operation, Voting, Multi-user-single-robot, MOSR, Remote robotics, Bayesian Analysis, Categorical Distributions, Weighted Network Optimization

1. INTRODUCTION

Multiple operator single robot systems (MOSR) differ from single operator single robot scenarios (SOSR) by allowing many teleoperators to simultaneously send commands to a vehicle in a far away location [Chong et al. (2000)]. The single operator single robot scenario is a straight forward problem incapable of confusion; no conflict resolution mechanisms or strategies are required since this is a one to one problem. MOSR systems on the other hand is a many to one problem which requires a more detailed analysis. Averaged multi-vector inputs [Goldberg and Chen (2001)], point-and-direct commands [Cannon (1992)], resource allocation problem [Song and Goldberg (2003)], numerical approach based on clustering and response time [Goldberg (2004)] and networked spatial dynamic voting strategy [Song (2009)] are some of the strategies that have been developed for simultaneous teleoperation of a single remote vehicle. The problem with the above strategies is that conflict resolution (in terms of voting or otherwise) always requires an additional human input. Network performance and teleoperator performance have no weight during conflict resolution.

Two major types[Saxena and Rai (2003)] of algorithms are used to implement simultaneous control: token based [Deldari (2007)], [Moallemi et al. (2006)] or permission based [Chaudhuri and Edward (2008)], [Harada and Yamashita (2004)]. In the former, a token is created and passed around, only an operator holding the token is allowed to command the robot which implies ofcourse that only one token can exist in a system at any point of time. The second class algorithms usually rely on the majority, so that the failure of one operator does not bring down the complete system. Permission based algorithms are subdivided into coterie based or voting based algorithms.

The size or scale of modern day applications is now causing more challenges to distributed MOSR systems. Voting systems have a distinct advantage in that they can adapt to dynamic changes[Ingols and Keidar (2001)] with inexpensive calculations/algorithms and can also scale to any number of operators.

The idea of this paper is to further investigate an intelligent and learning-capable dynamic process where teleoperation commands from operators are weighted with network parameters and operator performance to dynamically generate votes rated for effectiveness in order to enable collaborative teleoperation [Srinivasan et al. (2015)]. This algorithm uses bayesian predictive analysis and using an obstacle map calculates the prior and posterior of commands. Using this approach, the navigation task/command at hand is given priority rather than approaching it as a resource allocation problem.

2. WHY DO WE NEED SIMULTANEOUS OPERATION?

The department of Robotics and Telematics at the University of Wuerzburg is one of the few institutions across the world which has been offering a remote robotic platform for experiments on live hardware for over a decade now [Srinivasan and Schilling (2013)]]. The first setup was built with Java™Applets on the client side and connected to the server-side application using TCP to navigate a remote rover on the department's laboratory [Zeiger and Schilling (2005)] .

This, over the course of time, became technologically old and unstable and was replaced with a more modern setup including depth cameras installed for indoor laboratory operations (Fig: 1). The WebSocket protocol replaces TCP

2405-8963 © 2015, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2015.11.232

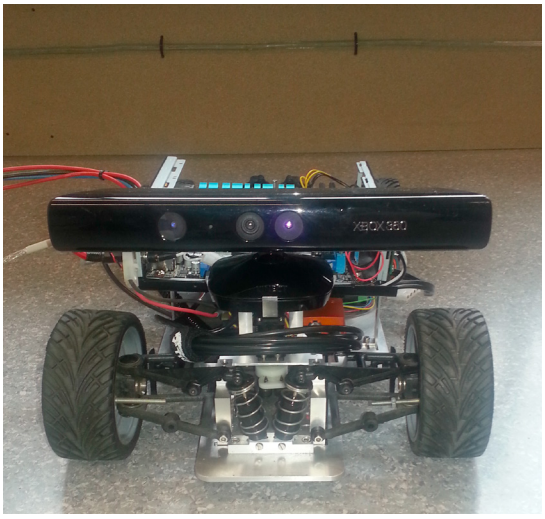

Fig. 1. The mobile robot used for teleoperation

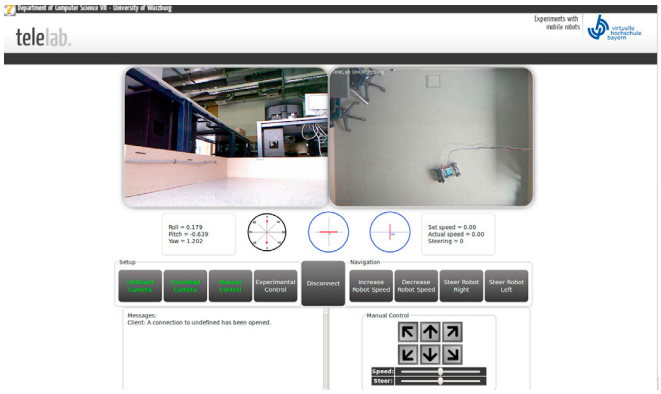

Fig. 2. Browser based teleoperation client interface [Srinivasan et al. (2015)]

and is used in browsers to communicate bidirectionally. This is different from pure TCP and varies from the more common HTTP-like request-response mechanism, XMLHttpRequests or long-hanging GETs.Srinivasan et al. (2013a) Dynamic user interfaces can now be developed and implemented using just JavaScript and HTML without any additional software [Srinivasan et al. (2013b)] Our teleoperation interface is one such as shown in Fig: 2

As student numbers began to increase, the major constraint of our platform was the time-availability of the robot per student. This SOSR system also implied that spatially seperated students could not work as a team during the actual teleoperation. This neccessitated a different approach leading us to questioning the existing setup and how to tweak it so that simultaneous operation can be enabled at the least cost.

3. VOTING

This section is a quick recap of the paper [Srinivasan et al. (2015)] which details the algorithm in more detail. The theory presented here is just a background for a reader to understand the system and it is not the goal of this paper to research voting algorithms. The main aim is to test a re-definition of what constitutes voting and to use this for collaborative teleoperation. Pessimistic or Optimistic algorithms are used in MOSR systems; where pessimistic algorithms bring consistency, but at the cost of availability. Static voting pattern is very common, an extremely simple majority based variation is described by [Thomas (1979)]. Permission to teleoperate is granted to a human operator if and only if the permission from a majority of n sites, that is, $[(n+1)/2]$, is obtained. This requires a worst case of $2n+2$ messages and in the best case scenario, $n/2+n+3$ messages [Thomas (1979)]. Majority voting is the most common in dynamic or static voting systems. Other variations such as relative consensus voting in which one parameter is prioritised and requests ordered accordingly [Cao et al. (2004)] or weighted voting where the sum of votes in agreement is atleast a majority [Gifford (1979)] are also available. Due to the lack of adaptability to dynamic changes in static voting algorithms, a state of simple minority prevents them from solving the decreased availability problem (often described as halted states [Thomas (1979)], [Barbara et al. (1989)]. Dynamic voting algorithms improve availability of the shared system in two ways - either by weighting the vote of an operator to be higher than the others or by ensuring service of non-critical tasks by the system in halted-states. In vogue dynamic voting systems are group-consensus [Barbara et al. (1989)], autonomous reassignment [Barbara et al. (1989)], dynamic linear [Jajodia and Mutchler (1990)], hybrid [Jajodia and Mutchler (1990)], [Jajodia and Mutchler (1989)] and dynamic weighted voting - most of them deriving from the majority principle in one way or the other.

Conflict resolution through voting mechanisms that have been implemented so far rely on a seperate track to determine operator or command precedence. The shared resource is not involved in this process at any stage. The main drawbacks of using such a system are

- It is an unnatural process for teleoperators,
- The whole process is resource expensive and
- the intelligence of the remote system where the actual sensor network is located is never used.

These drawbacks made us search for ways to improve a MOSR system which in turn lead us to think about what exactly constitutes a vote and how this process can be tweaked.

3.1 Can navigation commands be used as a vote?

Using the navigation control command as a vote negates the first two disadvantes listed above, namely more intuitive teleoperation and no parallel resource expensive voting system. The challenge is how best to incorporate a motion command in to a voting framework. This is where bayesian statistics, the heart of which is "given a set of data and a prior belief, a posterior belief can be computed" [Jackman (2009)] comes to the fore. An extension of this allows us to create various models that produce different future probabilities for a given set of date and this is extremely important to predict the most probable model given a set of evidence/data for which the Bayes factor will is used [Jeffreys (1950)]. Another major advantage of using Bayesian sequential analysis is that if a hypothesis is extremely unlikely a priori, even if the evidence suggests a possibility, it is rejected.

Download English Version:

https://daneshyari.com/en/article/710886

Download Persian Version:

https://daneshyari.com/article/710886

Daneshyari.com