Analyzing Teams of Cooperating Mobile Robots*

Bruce Randall Donald
James Jennings
Daniela Rus

TR 94-1429
June 1994

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

* Support for this work is provided in part by the NSF under grants No. IRI-9000532, IRI-9006137, by ONR under contracts No. N00014-92-J-1989 and N00014-92-J-39, by a Presidential Young Investigator award, and by the Air Force Office of Sponsored Research, the Mathematical Sciences Institute, Intel Corporation, and AT&T Bell Laboratories.

Analyzing Teams of Cooperating Mobile Robots[1]

Bruce Randall Donald James Jennings Daniela Rus
Robotics & Vision Laboratory
Computer Science Department
Cornell University
Ithaca, New York

Abstract

In [Don4], we described a manipulation task for cooperating mobile robots that can push large, heavy objects. There, we asked whether explicit local and global communication between the agents can be removed from a family of pushing protocols. In this paper, we answer in the affirmative. We do so by using the general methods of [Don4] analyzing *information invariants*. We discuss several measures for the information complexity of the task of pushing with cooperating mobile robots, and we present a methodology for creating new manipulation strategies out of existing ones. We develop and analyze synchronous and asynchronous manipulation protocols for a small team of cooperating mobile robots than can push large boxes. The protocols we describe have been implemented in several forms on the Cornell mobile robots in our laboratory.

1 Introduction

In this paper, we develop and analyze synchronous and asynchronous manipulation protocols for a small team of cooperating mobile robots than can push large boxes. The boxes are typically several robot diameters wide, and 1-2 times the mass of a single robot, although the robots have also pushed couches that are heavier (perhaps 2-4 times the mass, and 8×3 robot diameters in size). We build on the ground-breaking work of [Mason, EM] and others on planar sensorless manipulation. Our work differs from previous work on pushing in several ways. First, the robots and boxes are on a similar dynamic and spatial scale. Second, a single robot is not always strong enough to move the box by itself (specifically, its "strength" depends on the effective lever arm). Third, we do not assume the robots are globally coordinated and controlled. Fourth, our protocols assume neither that the robot has a geometric model of the box, nor that the first moment of the friction distribution is known. Instead, the robot combines sensorimotor experiments and manipulation strategies to infer the necessary information (the experiments have the flavor of [JR]). Finally, the pushing literature generally regards the "pushers" as moving kinematic constraints. In our case, because (i) there are at least two robot pushers and (ii) the robots are less massive than the box, the robots are really "force-appliers" in a system with significant friction. In this sense, our task is in some ways closer in flavor to *dynamic manipulation* [ML], even though the box dynamics are essentially quasi-static.

We develop a framework for analysis and synthesis, based on *information invariants* [Don4], to reveal these assumptions and expose the information structure of the task. We believe our theory has implications for the parallelization of manipulation tasks on spatially distributed teams of cooperating robots. To develop a parallel manipulation strategy, we start with a perfectly synchronous protocol (one with global coordination and control). Next, in distributing it among cooperating, spatially separated agents, we relax it to a MPMD[2] protocol with local communication and partial synchrony. Finally, we remove all explicit communication. The final protocols are essentially "uniform" in that the same program runs on each robot. However, the result is asynchronous, and hence it cannot be characterized as exclusively SPMD[3] nor MPMD. Ultimately, the robots must be viewed as communicating implicitly through the task dynamics, and this implicit communication confers a certain degree of synchrony on our protocols.

1.1 The Big Picture

Our goal here is to investigate the information requirements for the pushing tasks. This paper uses the theoretical framework and methodology introduced in [Don,Don4]. A central theme to previous work [Don] has been to determine what information is required to solve a task, and to direct a robot's actions to acquire that information to solve it. Key questions concern: (a) How much internal state should the robot retain? (b) How many cooperating agents are required, and how much communication between them is necessary? (c) How can the robot change (side-effect) the environment in order to record state or sensory information to perform a task? (d) How much information is provided by sensors? (e)

[1]Support for this work is provided in part by the NSF under grants No. IRI-8802390, IRI-9000532, IRI-9006137, by ONR under contracts No. N00014-92-J-1989 and N00014-92-J-39, by a Presidential Young Investigator award, and by the Air Force Office of Sponsored Research, the Mathematical Sciences Institute, Intel Corporation, and AT&T Bell laboratories.

[2]Multiple Program, Multiple Data

[3]Single Program, Multiple Data

How much computation is required by the robot? and (f) How much information is provided by the task mechanics?

These questions can be difficult. Structured environments, such as those found around industrial robots, contribute towards simplifying the robot's task because a great amount of information is encoded, often *implicitly*, into both the environment and the robot's control program. These encodings are difficult to measure. We wish to quantify the information encoded in the assumption that (say) the mechanics are quasi-static, or that the environment is not dynamic. In addition to determining how much "information" is encoded in the assumptions, we may ask the converse: how much "information" must the control system or planner compute? Successful manipulation strategies often exploit properties of the (external) physical world (e.g., compliance) to reduce uncertainty and hence gain information. Often, such strategies exploit mechanical computation, in which the mechanics of the task circumscribes the possible outcomes of an action by dint of physical laws. Executing such strategies may require little or no computation; in contrast, planning or simulating these strategies may be computationally expensive. Since during execution we may witness very little "computation" in the sense of "algorithm," traditional techniques from computer science have been difficult to apply in obtaining meaningful upper and lower bounds on the true task complexity. We hope that a theory of information invariants can be used to measure the sensitivity of plans to particular assumptions about the world, and to minimize those assumptions where possible.

In our quest for an intrinsic measure of the information requirements of a task, we are inspired by Erdmann's monograph on sensor design [Erd3], and the *information invariants* that Erdmann introduced to the robotics community in 1989 [Erd2]. We also observe that rigorous examples of information invariants can be found in the theoretical literature from as far back as 1978 (see, for example, [BK, Koz]). This work was motivated by the theoretical attack on perceptual equivalence begun by [DJ] and by the experimental studies of [JR]. Horswill [Hors] has developed a semantics for sensory systems that models and quantifies the kinds of assumptions a sensori-computational program makes about its environment. He also gives source-to-source transformations on sensori-computational "circuits." The paper [Don4] discusses the semantics of sensor systems. This formalism is used to explore some properties of what we call *situated sensor systems* and describes a way to transform sensori-computational systems. When one can be transformed into another, we say the latter can be "reduced" to the former, and we call the transformation a "reduction." We also derive algebraic algorithms for reducing one sensor to another. This machinery is only necessary if one wishes to automate the transformation process; it is quite easy to calculate reductions "by hand," using pencil and paper.

The goals outlined here are ambitious and we have only taken a small step towards them. There are several things we have learned. We can determine a lot about the information structure of a task by (i) parallelizing it and (ii) attempting to replace explicit communication with communication "through the world". Communication "through the world" takes place when a robot changes the environment and that change can be sensed by another robot. Here we give two different protocols (strategies) for a 2-robot pushing task: one protocol uses explicit communication and the other makes use of an encoding in the task mechanics of the same information. Our approach of quantifying the information complexity in the task mechanics involves viewing the world dynamics as a set of mechanically implemented "registers" and "data paths". This permits certain kinds of *de facto* communication between spatially separated robots. This "equivalence" of task mechanics and communication is operational in flavor, and we are still exploring its generality.

We believe that, by spatially distributing resources among collaborating agents, the information characteristics of a robot task are made explicit. That is, by asking, *How can this task be performed by a team of robots?* one may highlight the information structure. In robotics, the evidence for this is, so far, largely anecdotal. In computer science, one often learns a lot about the structure of an algorithmic problem by parallelizing it; we argue that a similar methodology is useful in robotics and illustrate our methodology for the pushing task.

1.2 Outline

We discuss questions (a) – (f) from Section 1.1 for an experiment with communicating robots. We foreground the task of pushing an object, using two communicating robots who need to infer the position of the first moment of the friction distribution with respect to their lines of pushing (see Figure 1a). We present, analyze and compare three pusing protocols using the tools introduced in [Don4]. We do so by rigorously comparing embedded sensori-computational systems, under the notion of *reduction*, which attempts to quantify when we can "efficiently" build one sensor out of another. We also sketch a general methodology for developing distributed manipulation protocols. Our methods generalize to other manipulation tasks and to larger teams of robots [DJR].

2 Pushing with Two Communicating Mobile Robots

2.1 Three Pushing Protocols

Consider the task whose goal is to push a box B in a straight line. Figure 1a depicts one robot (the reader should picture a robot manipulator, or gripper) executing this task. The robot consists of two rigidly connected fingers L and R; for example, they could be the fingers of a parallel-jaw gripper. One complication involves the micro-mechanical variations in the slip of the box on the table [Mas]. This phenomenon is very hard to model, and hence it is difficult to predict the results of a one-fingered push; we will only obtain a straight-line trajectory when the center of friction (COF) lies on the line of pushing. However, with a two-fingered push, the box will translate in a straight line so long as the COF lies between the fingers. An advantage of the two-finger pushing strategy is that the COF can move some and the fingers can keep pushing, since we only need ensure the COF lies in some region C (see Figure 1a), instead of on a line. If the COF

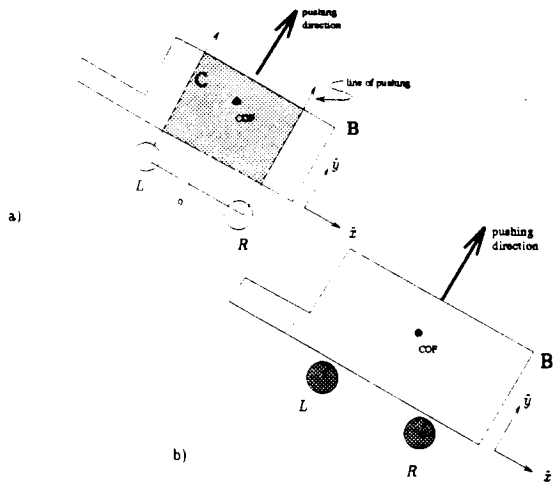

Figure 1: (a) the "two-finger" pushing task vs. (b) the two robot pushing task. The goal is to push the block B in a straight line.

moves outside C, then the fingers can move sideways to "capture" it again. We have implemented the control loop described as Protocol O on our PUMA (see Figure 2). The basic idea is to sense the reaction torque τ about the point 0 in Figure 1a. If $\tau = 0$, push forward in direction $\hat{y}$. If $\tau < 0$ move the fingers in $\hat{x}$; else move the fingers in $-\hat{x}$.

```
Repeat :   measure(τ)
           case  (τ = 0) ⇒
                   push(p)
                 (τ < 0) ⇒
                   move(R)
                 (τ > 0) ⇒
                   move(L)
```

Figure 2: Protocol O (for a two-fingered gripper).

From the mechanics perspective it might appear we are done. However, it is difficult to overstate how critically Protocol O above relies on global communication and control. Now, consider the analogous pushing task in Figure 1b, in which mobile robots perform the pushing. The robots in our lab are autonomous and equipped with a ring of 12 simple Polaroid ultrasonic sonar sensors. Each robot has onboard processors for control and programming. We equip each robot with 12 infra-red modems/sensors, arrayed in a ring about the robot body. Each modem consists of an emitter-detector pair. In addition, each robot has a ring of one-bit contact ("bump") sensors.

We assume the following: (1) robots can sense the relative orientation of objects with which they are in contact [JR]; (2) robots know that they are on the same flat face of the object; (3) both robots know the direction of pushing, p; and (4) robots can synchronize their velocities.

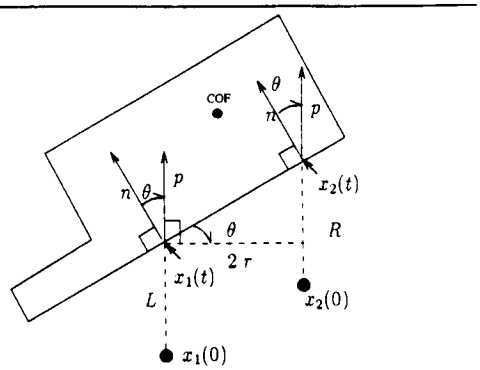

Figure 4: Protocol I(QS). The normal to the box face is denoted by n. The x axis is parallel to the direction of pushing p. The lines of pushing are distance $2r$ apart (perpendicular distance). θ is the angle between n and p.

The pushing strategy described as Protocol O can be approximated by observing the following (see Figures 1b, 3). Each robot can measure its applied force (f_1 or f_2) and communicate this data to the other. This allows the robots to compute the net torque τ about a point in between the two robots, and from the sign of this quantity, to infer the location of the first moment of the friction distribution of the box. If the first moment of the friction distribution is between the two lines of pushing, each robot continues pushing alone the line p. If there is a positive net torque, the instantaneous center of friction is to the left of both robots.. In this situation, the left robot is in contact with the box, while the right robot may or may not be in contact. The left robot can "recapture" the center of friction between the two lines of pushing by moving left along the face of the box (move(L)). If the right robot is not in contact with the box (the predicate (**break?**) returns TRUE) it executes a guarded-move (motion until contact) in the direction p. Otherwise this robot takes the null action, $\emptyset$. The case when the net torque is negative is symmetric. We call this Protocol I (see Figures 1b, 3).

A variant of this protocol can be derived for a quasi-static (QS) system. Here, relative displacements along the line of pushing p are measured instead of forces. Figure 4 shows a configuration where the two robots originate at positions $x_1(0)$ and $x_2(0)$, respectively. Their locations at time t are $x_1(t)$ and $x_2(t)$. In this protocol (Figure 5), the initial locations of the robots are communicated to determine their offset Σ_0. Σ_0 "specifies" (or better, *parameterizes*) the pushing task: this offset determines the pushing direction p relative to the initial orientation of the box face. The robots exchange location information successively at each loop iteration; this information

Left Robot	Communication	Right Robot
Repeat : measure(f_1)		Repeat : measure(f_2)
	$\boxed{L \rightarrow R}$ (f_1)	
		$\tau = r_1 \times f_1 - r_2 \times f_2$
	$\boxed{L \leftarrow R}$ (sign(τ))	
case ($\tau = 0$) $\Rightarrow$ push(p) ($\tau < 0$) $\Rightarrow$ if(**break?**) **guarded-move**(p) else $\emptyset$ ($\tau > 0$) $\Rightarrow$ move(L)		case ($\tau = 0$) $\Rightarrow$ push(p) ($\tau > 0$) $\Rightarrow$ if(**break?**) **guarded-move**(p) else $\emptyset$ ($\tau < 0$) $\Rightarrow$ move(R)

Figure 3: Protocol I

is used to infer the direction of motion of the box. We call this Protocol I(QS) (see Figures 4, 5).

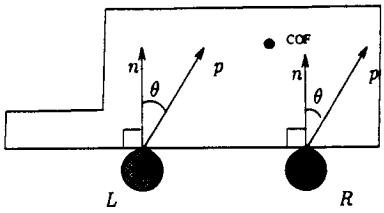

Figure 6: Protocol II.

We now derive a different version of Protocol I(QS) by observing that the information needed to determine the motion of the box (*i.e.* Σ_0 and Σ) is related to the angle θ between the normal to the face of the box n and the direction of pushing p as follows: $2r\tan\theta_0 = \Sigma_0$ and $2r\tan\theta = \Sigma$ (see Figures 4, 6, 7). Moreover, we observe that the tangent function is monotonic and sign preserving; this means we can adapt the control system to servo on θ instead of Σ, without knowing r. Specifically, the robots measure θ_0 (the initial angle between n and p; see [JR]), and compare this value to the angle $\theta(t)$ measured at time t in order to infer the direction of motion of the box. The sign of the change in this angle defines the sense of the rotation of the box. The robots adjust their pushing location on the face of the box accordingly. This is an example of how the robots can use the task dynamics instead of explicit communication to determine their next actions. We call this pushing strategy Protocol II (Figure 7).

2.2 Comparing the Protocols

Now, we ask, how do protocols I, I(QS), and II compare to one another with respect to the questions (a) – (f) posed above? We first note that the three protocols require different sensing capabilities. Protocol I relies on force sensing, Protocol I(QS) relies on position sensing, and Protocol II relies on orientation sensing. Next we observe that the robots must coordinate to find locations that result in a stable pushing along p. This coordination is accomplished differently in the three protocols. In Protocol I and Protocol I(QS) the robots synchronize by exchanging their sensed values. Robots executing Protocol I require communicating $\log f_1$ bits to transmit the value of force f_1, and 2 bits to transmit the sign of the torque τ. Robots executing Protocol I(QS) require $\log x_1$ bits to transmit the value of the distance x_1, and 2 bits to transmit the sign of s. In Protocol II there is no explicit communication and the synchronization is realized through the world, by monitoring the change in the angle θ between the normal to the face and the pushing direction. In other words, the robots infer the motion of the object by decoding changes in the task mechanics. Thus, protocols I and I(QS) rely on direct communication, while protocol II does not. The internal state requirements of the three strategies are also different. Protocol I requires no internal state. Protocol I(QS) requires a register to record the value Σ_0. Protocol II requires a register to record the value θ_0.

We can get a deeper understanding of the relationship between these protocols by attempting to "transform" or "reduce" one to the other. We do so below.

3 Reductions and Transformations

We present here a very brief summary of our model of sensori-computational systems, which we view as "circuits." (See [Don4] for a full treatment of these concepts.) We model the circuits as graphs. Vertices correspond to

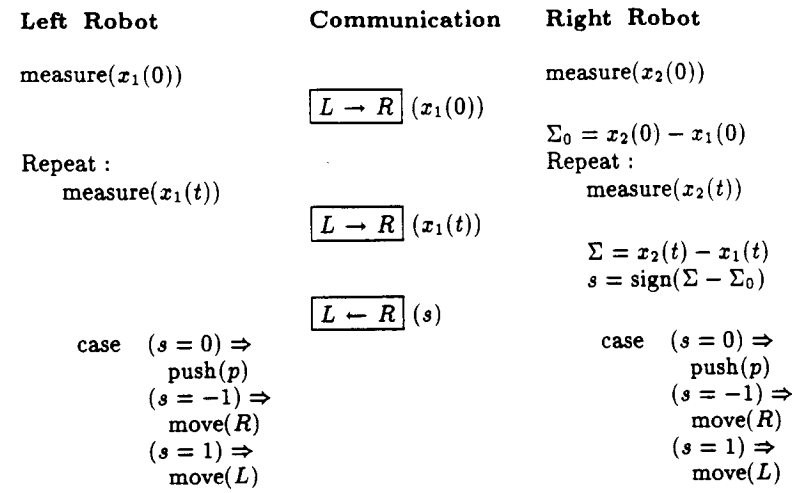

Figure 5: Protocol I (Quasi-Static). The case for breaking contact is not shown; it can be handled as in Figure 3.

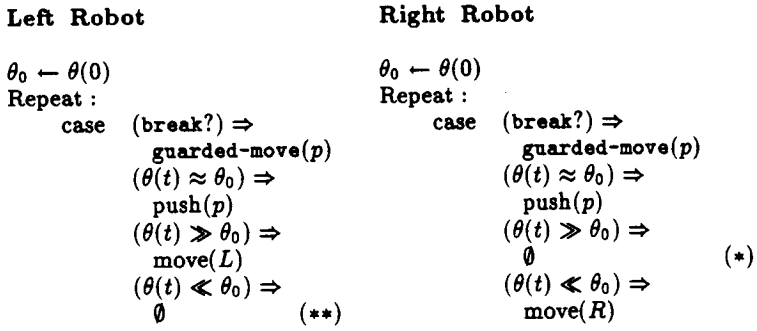

Figure 7: Protocol II. This protocol is "almost uniform," and can be made uniform by changing the $\emptyset$ lines (*) to MOVE(L) and (**) to MOVE(R). Note that "uniform" does not quite imply SPMD, since the protocols run asynchronously.

different sensori-computational components of the system (what we will call "resources" below). Edges correspond to "data paths" through which information passes. Different immersions of these graphs correspond to different spatial allocation of the "resources." We also define an operator + as a way to "combine" sensori-computational systems. Below we use the term *"sensor system"* to mean *"sensori-computational system"* where it is mellifluous.

3.1 Situated Sensor Systems

Definition 3.1 *A* labelled graph *$\mathcal{G}$ is a directed graph (V, E) with vertices V and edges E, together with a labelling function that assigns a label to each vertex and edge. Where there is no ambiguity, we denote the labelling function by ℓ.*

Definition 3.2 *A* sensor system *S is represented by a labelled graph (V, E). Each vertex is labelled with a* component. *Each edge is labelled with a* connection.

See Figures 8-9 for illustrations for the circuits—that is, the sensor systems for protocols I (QS) and II. We will use the terms *protocol* to refer to the computer programs in Figures 5-7, and *circuit* for the sensor systems in Figures 8-9. In the figures, the components correspond to boxes: ODOM is an odometer, the "signal"[4] coming out of this box is the odometry reading. The box $-$ performs subtraction, the boxes $x_1(0)$ and Σ_0 are registers, and they implement internal state. The part of the circuit labelled case interfaces to the control part of the circuits, which is the same for both protocols. For technical reasons, we define a resource called "output." Each sensor system must have exactly one vertex with this label. The output vertex of the sensor system is where the output of the sensor is measured.

[4] We use *"signal"* as a metaphor; our circuits are strictly digital. Either *message* or *stream* would be better, but both have distracting religious connotations.

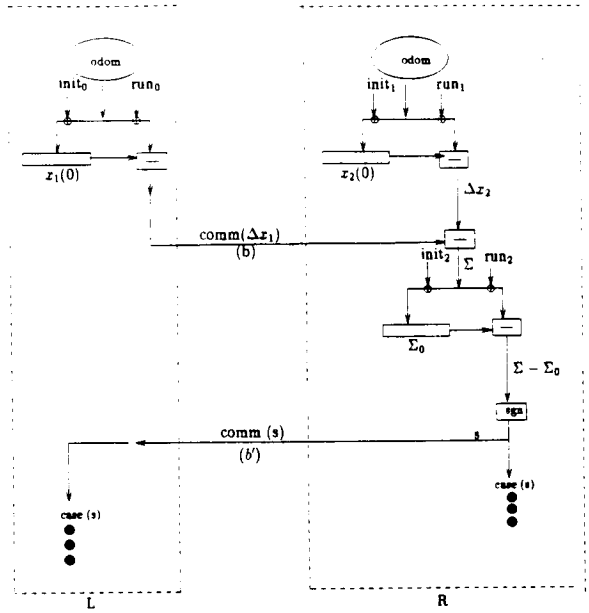

Figure 8: Sensor system P.I(QS), a circuit for Protocol I (QS). This circuit shows one possible implementation of the protocol. Figures 8-9 do not show how to handle lost of contact (i.e., the (break?) case), but this circuitry is easily added, and is the same for both P.I(QS) and P.II.

INIT is one bit of state, and RUN= $\overline{\text{INIT}}$. The small crossed circles ($\oplus$) that these bits run into are gates; the $\downarrow$ input must be 1 for the $\leftrightarrow$ signal to pass.

Connections are like data-paths in that they carry information; a connection's label represents the information that will be sent along that path. We adopt the convention that two components can communicate without an (explicit) connection when they are spatially colocated.

Consider a sensor system with vertices V. For each vertex v in V, we assume there is a configuration space C. A point in this space C represents the configuration of the component.

Definition 3.3 *A* situated *(or* immersed) sensor system *S is a sensor system $S = (V, E)$, together with an immersion $\phi : V \to C$ of the vertices. If $v \in V$, then we call $\phi(v)$ the* configuration of the vertex *v. When there is no ambiguity, we also call $\phi(v)$ the* configuration of the component *$\ell(v)$.*

We can now define what it means for two systems to be equivalent:

Definition 3.4 *Given two sensor systems S and Q, we say Q* simulates *S if the output of Q is the same as the output of S. In this case we write $S \cong Q$. More generally, suppose we write $(S, \phi) \cong (\mathcal{U}, \psi)$ for two situated sensor systems. This is clearly well-defined when ϕ and ψ are total. Now, suppose that ϕ and ψ are partial, leaving unspecified the configurations of components $\ell(v)$ of S and $\ell(u)$ of $\mathcal{U}$. Then the definition is taken to mean that $(\mathcal{U}, \psi)$ simulates (S, ϕ) for* any *configuration of v and u.*

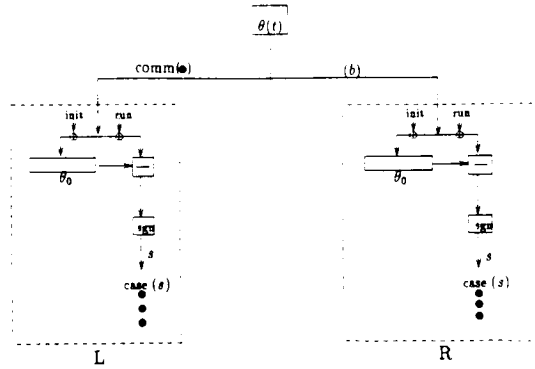

Figure 9: Sensor system P.II, a circuit for Protocol II.

3.2 Permutation

Definition 3.5 *Let $\mathbb{S} = (S, \phi)$ be a situated sensor system. A* permutation *$\mathbb{S}^*$ of $\mathbb{S}$ is a situated sensor system (S, ϕ^*) such that the domain $\phi^{-1}C$ of ϕ and the domain $\phi^{*-1}C$ of ϕ^* are the same.*

We can also consider an alternate model, called *edge permutation*, where the edge connectivity changes. Consider a graph with vertices V and edges E. Start with any bijection $\sigma : V^2 \to V^2$. We call σ an *edge permutation*, since it induces the restriction map $\sigma_{|E} : E \to \sigma(E)$ on the edge set E. In this paper we will restrict our edge permutations to *class* edge permutation, in which we segregate edges into two classes, and permute only within a class.[5] The two classes, *internal* and *external*, correspond to communication within and between (respectively) particular subcircuits. A subcircuit may represent, e.g. a single mobile robot.

We define a *graph permutation* to be a vertex permutation followed by an edge permutation.

Let $(\mathcal{U}, \phi)$ be a situated sensor system. A graph permutation of $\mathcal{U}$ is given by $\mathcal{U}^* = (\mathcal{U}, (\phi^*, \sigma))$ where ϕ^* is a vertex permutation, and σ is an edge permutation.

3.3 Combination

Definition 3.6 *Consider two graphs $\mathcal{G} = (V, E)$ and $\mathcal{G}' = (V', E')$. We define the* combination *$\mathcal{G} + \mathcal{G}'$ of $\mathcal{G}$ and $\mathcal{G}'$ to be $\mathcal{G} + \mathcal{G}' = (V \cup V', E \cup E')$.*

We may define + on sensor systems (Definition 3.2) by lifting the definition for graphs. We may define + on two immersed graphs whenever the immersions are *compatible*. An immersion ϕ of $\mathcal{G}$ and an immersion ψ of $\mathcal{G}'$ are said to be *compatible* when the two immersions agree

[5] Class edge permutation leaves unchanged the complexity bounds and the lemmas of [Don4].

on the intersection $V \cap V'$ (for total immersions) or more generally, on $\phi^{-1}C \cap \psi^{-1}C$ (for partial functions). The definition of $-$ is analagous.

3.4 Reductions, Calibration, and Codesignation

As we observed in [Don], calibration exploits external state. *Calibration complexity*, defined formally in [Don4], measures how much information we add to a sensor system when we install and calibrate it. Installing a sensor system may require physically establishing some spatial relation between two components of the system. In this case we say the two components *codesignate* by the spatial relation. More generally, we may have to establish a relation between a component and a reference frame in the world.

Definition 3.7 *We write $S \leq Q$ when (1) Q simulates S ($S \cong Q$), (2) S dominates Q in calibration complexity, and (3) the maximum bandwidth of S is at least that of Q (def. 3.9 below).*

Definition 3.8 *We write $S \leq^* Q$ if there exists a (graph) permutation Q^* of sensor system Q such that $S \leq Q^*$.*

3.5 Reductions using Communication

Definition 3.9 *We define the* internal *(resp.,* external*)* bandwidth *of a sensor system S to be the greatest bandwidth of any internal (resp., external) edge in S. The* raw output size *of S is the size of the value it outputs. We define the* maximum bandwidth *of S to be the greatest of the internal bandwidth, external bandwidth, and the raw output size of S.*

We define $\text{COMM}(S)$ to be $\text{COMM}(2^k)$ where k is the *maximum bandwidth* (Definition 3.9) of S; we treat k as an upper bound on *relative intrinsic output complexity* of S.

Definition 3.10 *Consider two sensor systems S and Q. We say S is* efficiently reducible *to Q if*

$$S \leq^* Q + \text{COMM}(S).$$

In this case we write $S \leq_1 Q$.

The sensor system $(Q + \text{COMM}(S))^*$ from the definition may be implemented as the sensor system Q permuted in an arbitrary way, plus one extra data path whose bandwidth is that of the largest flow in S.[6]

The reduction $\leq_1$ (Definition 3.10) is a "1-wire" reduction. It does not appear to be transitive. The reduction $\leq^*$ (Definition 3.8) is a *"0-wire"* reduction. It is transitive for simple sensor systems [Don4]. We could analogously define a *k-wire reduction* $\leq_k$ by modifying the equation in definition 3.10 to read $S \leq^* Q + k \cdot \text{COMM}(S)$, where $k \cdot \text{COMM}(S)$ denotes $\overbrace{\text{COMM}(S) + \cdots + \text{COMM}(S)}^{k \text{ times}}$.

[6] See proof in [Don4].

4 Comparing Protocols Using Reductions

We now apply the ideas above to compare our protocols, P.I(QS) and P.II (the circuits in Figures 5–7).

Theorem 4.1 *Let* P.II, P.I(QS), ODOM, *and θ be the sensor systems defined as above. Then,*

$$\text{P.II} \leq_k \text{P.I(QS)} - 2\text{ODOM} + \theta \tag{1}$$

for $k = 1$. Moreover, eq. (1) does not hold for $k = 0$.

Proof: Consider the sensor system $\mathcal{U}$ obtained by removing both odometers from circuit P.I(QS), and adding a θ-source:

$$\mathcal{U} = \text{P.I(QS)} - 2\text{ODOM} + \theta. \tag{2}$$

Now, consider permutations of $\mathcal{U}$, and recall Definition 3.8. We first ask whether P.II can be reduced to $\mathcal{U}$ using $\leq^*$. That is, P.II $\leq^* \mathcal{U}$? First, we note that we can move around the registers and $\boxed{-}$'s from $\mathcal{U}$ to situate all the components of P.II. We also have some leftover components (and wires). P.II requires two sign boxes; however, the $\boxed{\text{SGN}}$ box just selects out the sign bit. To build the extra sign box we can just ignore the other bits, or we can use the leftover hardware from $\mathcal{U}$ to build a small circuit to simulate $\boxed{\text{SGN}}$. We need to argue that a register big enough to hold $x_i(0)$ will also hold θ_0; this follows from $2r \tan \theta(t) = \Sigma(t)$, or from "decalibration" [Hors]. Next, we see that we can permute the internal edges of $\mathcal{U}$ to wire up the components of P.II *in situ*—internally. What about externally?

Permuting the external wiring almost works, but not quite. $\mathcal{U}$ has two external data paths, (b′) and (b), with bandwidth 2 bits and $\log \Delta x_1$ bits (resp) (Figure 8). Now, since we only allow class edge permutation (as in Section 3.2), we must permute external edges to external edges and internal edges to internal edges. Therefore, in Figure 9, the edge (b) from $\mathcal{U}$ will suffice as a datapath from $\theta(t)$ to R, since it has adequate bandwidth. However, the datapath (b′) from $\mathcal{U}$ does not have adequate bandwidth to carry information from $\theta(t)$ to L. In order to build P.II from $\mathcal{U}$, we must add another external data path $\text{COMM}(\cdot)$ from $\theta(t)$ to L. Now, what is the argument to $\text{COMM}(\cdot)$? This data path must have bandwidth of at least the relative intrinsic output complexity of P.II, or $\log \Delta\theta$ bits. Hence we may parameterize this new edge by writing $\text{COMM}(\text{P.II})$, following Section 3.5. Hence, we see that

$$\text{P.II} \leq (\mathcal{U} + \text{COMM}(\text{P.II}))^*. \tag{3}$$

Therefore by Definition 3.8,

$$\text{P.II} \leq^* \mathcal{U} + \text{COMM}(\text{P.II}), \tag{4}$$

so using Definition 3.10, we have P.II $\leq_1 \mathcal{U}$. Hence we conclude

$$\text{P.II} \leq_1 \text{P.I(QS)} - 2\text{ODOM} + \theta, \tag{5}$$

which implies eq. (1) as desired. □

This formalizes our intuition that, by removing odometry but adding relative normal sensing, we can accomplish the pushing task without explicit communication. More precisely, we show how to build one circuit P.II "efficiently" out of the other ($\mathcal{U}$). To transform P.I(QS) into P.II, the operators + and − quantify what resources we add and delete. Relative information complexity allows us to measure the effective communication "through the world." The permutation quantifies the redistribution of resources.

5 Computing Reductions

Theorem 4.1 is a proof done "by hand." That is, we can in principle determine that eq. (1) holds (for $k > 0$) by showing—"by hand"—the existence of a suitable permutation. It is somewhat surprising that we can in fact automate this process: [Don4] gives algorithms for deciding the relation $\leq_1$. More precisely, given suitable encodings of two sensor systems $\mathcal{S}$ and $\mathcal{U}$, we can computationally decide whether $\mathcal{S} \leq_1 \mathcal{U}$ [Don4]. The algorithm is too complicated to describe here. The basic idea involves employing the theory of real closed fields with bounded quantification.

6 Conclusion

We hope that by analyzing the manipulation protocols presented in this paper, we have been able to reveal the information structure of the cooperative pushing task. In our laboratory, multiple mobile robots are executing these strategies and others for both pushing and reorientation of large objects [DJR]. In analyzing the results of our experiments, we have demonstrated precise equivalences between different types of sensing, communication, and internal and external state, by reducing one sensor system to another; the nature of these equivalences is conditioned by the task.

We may ask at this time any number of specific questions about the theory of information invariants and the application of that theory. For example, can we record "programs" in the world in the same way we may externalize state? Is there a "universal" manipulation circuit which can read these programs and perform the correct strategy to accomplish a task? Such a mechanism might lead to a robot which could infer the correct manipulation action by performing sensori-motor experiments, thus obviating the need for the specific protocols we have analyzed here.

We think that information invariants can serve as a framework in which to measure the capabilities of robot systems, to quantify their power, and to reduce their fragility with respect to assumptions that are engineered into the control system or the environment. We believe that the equivalences that can be derived between communication, internal state, external state, computation, and sensors, can prove valuable in determining what information is required to solve a task, and how to direct a robot's actions to acquire that information to solve it.

Bibliography

[BK] **Blum, M. and Kozen, D.** *On the power of the compass (or, why mazes are easier to search than graphs),* Proc. 19th Symp. Found. Computer Science, Ann Arbor, MI, pp. 132-42 (1978).

[Don] **Donald, B. R.** *Information Invariants in Robotics, Parts I and II,* IEEE International Conference on Robotics and Automation, Atlanta, GA. (1993).

[Don4] **Donald, B. R.** *On Information Invariants in Robotics,* Cornell Computer Science Department Technical Report TR 93-1341. (1993).

[DJ] **Donald, B. R. and J. Jennings** *Constructive Recognizability for Task-Directed Robot Programming,* Jour. Robotics and Autonomous Systems, **(9)**, No. 1, Elsevier/North-Holland pp. 41-74. (1992).

[DJR] **Donald, B. R., J. Jennings, and D. Rus** *Experimental Information Invariants for Cooperating Autonomous Mobile Robots,* International Joint Conference on Artificial Intelligence (IJCAI) Workshop on Dynamically Interacting Robots. Chambery, France (Aug 28) (1993).

[Erd2] **Erdmann, M.** *On Probabilistic Strategies for Robot Tasks,* Ph.D. thesis, MIT Department of EECS, MIT A.I. Lab, Cambridge MIT-AI-TR 1155 (1989).

[Erd3] **Erdmann, M.** *Action Subservient Sensing and Design,* IEEE ICRA, Atlanta. See also the Carnegie-Mellon report CMU-CS-92-116. (1993).

[EM] **Erdmann, M., and M. Mason**, "An Exploration of Sensorless Manipulation", *IEEE International Conference on Robotics and Automation*, San Francisco, April, 1986.

[Gri] **Grigoryev D. Yu.**, "Complexity of Deciding Tarski Algebra" Jour. Symbolic Computation, **5**, (1), (1988), pp. 65–108.

[Hors] **Horswill, I.** *Analysis of Adaptation and Environment,* Submitted to *Artificial Intelligence* (1992).

[JR] **Jennings, J. and Rus, D.** *Active Model Acquisition for Near-Sensorless Manipulation with Mobile Robots,* The IASTED International Conference on Robotics and Manufacturing, Oxford, UK (1993).

[Koz] **Kozen, D.** *Automata and Planar Graphs,* Fundamentals of Computing Theory, Proc. Conference on Algebraic, Arithmetic, and categorical methods in Computation Theory (Berlin) ed. L. Budach, Akademie Verlag (1979).

[Mas] **Mason, M. T.** 1986. Mechanics and Planning of Manipulator Pushing Operations. *International Journal of Robotics Research* 5(3).

[ML] **Mason, M. and Lynch, K.** *Dynamic Manipulation,* Proc. of the 1993 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Yokohama, Japan, July 26-30 (1993).

[Rus] **Rus, D.** "Fine Motion Planning for Dexterous Manipulation", PhD. Thesis available as CU-CS-TR 92-1323 (August) from Comp. Sci. Dept., Cornell University, 1992.

Acknowledgment. Many key ideas in this paper arose in discussions with Tomás Lozano-Pérez, Mike Erdmann, Matt Mason, and Ronitt Rubinfeld. The robots and experimental devices described herein were built in our lab by Jim Jennings, Russell Brown, Jonathan Rees, Craig Becker, Mark Battisti, and Kevin Newman; these ideas could never have come to light without their help. Debbie Lee Smith, Amy Briggs, and Karl-Freidrich Böhringer made suggestions and helped draw the other figures, and we are very grateful to them for their help.