Moving Furniture with Teams of Autonomous Robots

Daniela Rus
Department of Computer Science
Robotics Laboratory
Dartmouth College
Hanover, NH 03755

Bruce Donald Jim Jennings
Department of Computer Science
Robotics and Vision Laboratory
Cornell University
Ithaca, NY 14853

Abstract

We wish to organize furniture in a room with a team of robots that can push objects. We show how coordinated pushing by robots can change the pose (position and orientation) of objects and then we ask whether planning, global control, and explicit communication are necessary for cooperatively changing the pose of objects. We answer in the negative and present, as witnesses, four cooperative manipulation protocols that use different amount of state, sensing, and communication. We analyze these protocols in the information invariant framework. We formalize the notion of resource tradeoffs for robot protocols and give the tradeoffs for the specific protocols discussed here.

1 Introduction

We wish for robots to perform sophisticated tasks in structured environments, such as an assembly line, as well as unstructured ones, such as space, undersea, or outdoors. Such robots must have the ability to manipulate objects. Consider a manipulation task in which a couch is to be pushed out of a room. Suppose the length of the couch is too wide to go through the door, but its side is narrow enough. The couch needs to be rotated before it can be moved out. We wish to do this task with a team of mobile robots that can push objects. The objects vary in size and shape and the robots do not have geometrical models of the shape nor mechanical models of their frictional or mass properties. The robots and objects are on a similar dynamic and spatial scale, so a single robot is not always strong enough to move an object by itself. In this paper, we develop, analyze, and contrast both synchronous and asynchronous manipulation protocols for a small team of mobile robots that can affect changes in the orientation of large objects (*e.g.* couches) as part of a furniture moving task.

The *reorientation problem* is to affect a desired change in the orientation of an object. We are only interested in distributed, cooperative solutions to the reorientation problem. We differentiate between the properties of manipulation protocols by robots according to the following hierarchy. In a *manipulation task*, robots use forces to reorganize the robots' space. In a *parallel manipulation protocol*, two or more robots apply forces to the same coupled dynamical system. In a *distributed manipulation protocol*, the computation is distributed among the robots in a way that qualifies as a distributed computation. Finally, Brown defines a *strongly cooperative* manipulation task as one that can only be performed by parallel manipulation protocols [Bro]. Hence, we present strongly cooperative protocols for the reorientation problem.

Our research agenda is to investigate the information requirements for the reorientation task. This paper uses the theoretical framework of [Don] and the results and experiments have the flavor of [DJR]. Specifically, we treat manipulation from the minimalist stance [EM, Erd2, BBDJR] and we ask: *(i) Can we do reorientations without communication? (ii) Can we do reorientations without planning? (iii) Can we do reorientations without global control? (iv) Can we do reorientations without geometric models?* In [DJR] we show that we can answer questions (i)–(iv) affirmatively for the task of pushing an object along a straight line with multiple robots. Posing (i)–(iv) for the reorientation task is much more challenging. (For example, we do not believe that cooperative reorientation can be done with a potential field method.) In this paper we answer questions (i)–(iv) affirmatively for the reorientation task. The two skills, pushing and reorienting, when interleaved, suffice for many manipulation tasks.

Perhaps our most surprising result is contained in the asynchronous, uniform (SPMD[1]), and parallel protocol for cooperative reorientation that requires no explicit communication between the robots, no world models, no planner, and is fully parallel (SPMD). We derived this system by first postulating a system in which the robots rely on communication to synchronize their actions. We then postulated transforming this into a second, also hypothetical system: the transformation entailed replacing the explicit communication by reads and writes into a shared data structure. Finally a third system was designed by implementing the shared data structure as a real-valued "mechanical register"[2] [DJR]. The last transformation implements implicit communication through the world (through the task mechanics.) While many researchers have proposed doing multiple mobile robot projects without direct communication, the only previous system

[1] SPMD = *S̲ingle P̲rogram M̲ultiple D̲ata.*

[2] Informally one should imagine the "mechanical register" as the state (pose and velocity) of the single couch which the mobile robots jointly manipulate, in parallel.

that has been successful in achieving this is [BHD], who treat a foraging task[3].

In our lab we implemented and tested the final protocol. The intermediate, unimplemented protocols are hypothetical design constructs (*i.e.*, parallel algorithms) that emerge from our methodology. At each step the transformations above are formalized using *information invariants,* [Don], which quantify resource trade-offs. These trade-offs confess what resources (sensors, actuators, computation, communication, *etc.*) are added and deleted at each step. Furthermore, information invariants measure the resource distribution and the power thereby obtained. By confessing each trade-off a robot designer can decide whether (for example) eliminating a planner is worth the additional cost of adding sensors, processing, and communication that are needed in compensation. More strongly, we believe that a parallel algorithm that eliminates a major resource (such as planning) without confessing the additional resources required in compensation is not desirable. Finally, by careful design and the use of the information invariants formalism we can obtain simple protocols that exhibit coordinated, cooperative performance using a near minimum amount of resources. The analysis we advocate is complicated, but the final result is a simple protocol that use a near-optimal resource signature.

1.1 Previous work

There has been a lot of work on cooperative systems of robots. For an excellent review see [CFKM]. Previous work on cooperative manipulation has focused on pushing. Cooperative pushing has been treated in the context of box-pushing with multiple robots. In [Par] a reinforcement learning strategy that focuses on task allocation is presented.

We build on the ground-breaking work of [Mas1, EM] and others on planar sensorless manipulation. Inspired by [EM, Erd2] we focus on developing near-sensorless (*i.e.*, that require only sparse sensing) manipulation protocols [Rus, JR, DJR]. Our previous work on cooperative manipulation treats pushing a large object from a minimalistic viewpoint[DJR]. We present and analyze four pushing protocols that require different amount of communication and sensing. The final pushing protocol requires no explicit communication between the robots. Our previous work also considers the reorientation problem. In [Rus, BR] we present and analyze an off-line reorientation algorithm.

2 Reorienting Large Objects with Autonomous Mobile Robots

Consider the task whose goal is to change the orientation of a large object by a given amount. This is called the *reorientation* task. Figure 1 depicts one robot reorienting a large object. A robot can generate a rotation by applying a force that is displaced from the center of friction. This property relates the dynamics and the geometry or reorientations [Mas1] and it can be used to effect continuous reorientations with mobile robots. The idea is to compliantly apply a sliding force on the face of the object[4]. We call this action a *pushing-tracking* step. When the end of the face is reached, the robot may turn around to reacquire contact and repeat the pushing-tracking. A robot that has gone past the end of a face effectively losing contact with the object has *broken contact* with the object. A robot whose maximum applied force (defined by a threshold) does not change the pose of the object has encountered an *impediment.*

One robot may effect any desired reorientation by repeated pushing-tracking steps if it can apply a large enough force, but it may require a large workspace area for the rotation. We are interested in strategies for reorienting objects *in place*. This can be done by a team of k robots (see Figure 1(b).) The k grasping robots can simultaneously constrain the motion of the object and execute *pushing-tracking* operations. We can measure the degree of parallelism in the reorientation algorithm by counting how many of the robots are *active*, *i.e.*, that execute push-tracking motions, and how many of the robots are *stationary*, *i.e.*, that constrain the motion of the object by staying fixed in place. We show how to select the active and stationary robots and how to decide on role-switching over the course of an algorithm.

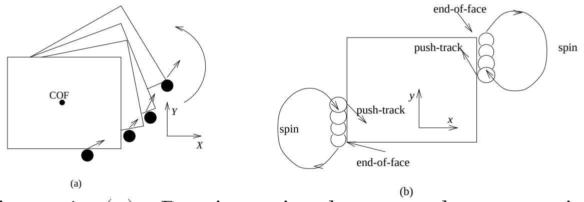

Figure 1: (a): Reorientation by one robot executing pushing-tracking. (b) A system of two robots reorienting a couch. The robot motions are shown in an object-fixed frame. Each robot executes a pushing-tracking motion. Robots recognize when they reach the end of a face by breaking contact, and execute a spinning motion to turn around and reacquire contact.

We now present four different but "equivalent" reorientation protocols that have different degrees of parallelism, synchrony, and resource utilization. Our notion of "equivalence" comes from looking at the task as a dynamical system. Consider the configuration space $\mathcal{C}$ of the manipulated couch. We call two manipulation protocols *equivalent* if the protocols have the same forward-attracting compact limit set in $\mathcal{C}$ [Don] (pg. 284) and [DJR]. All our protocols rely on the ability of robots to execute *push-tracking* motions.

2.1 An Off-line, Global Control Protocol

The off-line global control strategy denoted by GLOBAL-OFFLINE requires three robots and is described and analyzed in detail in [Rus]. The algorithm

[3] A team of robots wander in search for "food" that is randomly distributed throughout their workspace.

[4] This strategy can be implemented by a force that forms an acute angle on the contacting edge. This is similar to hybrid control [RC] which would be used for a dexterous hand [Rus].

consists of a sequence of pushing-tracking steps, where one robot is active and two robots are stationary at all times (see Figure 2.) A global controller sends and receives control signals to the robots. Under the assumption that the robots are already in contact with the object, the algorithm can be summarized as follows:

- Active robot:
  1. push-track until position or force termination
  2. signal the global controller
  3. become stationary
- Stationary robots:
  1. sense for relative motion (*slip*)
  2. if no slip, signal the global controller
  3. when signal, become active

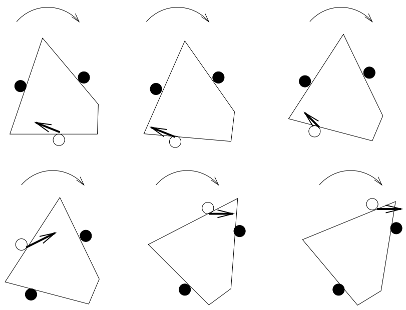

Figure 2: A two-step pushing-tracking sequence of snapshots for reorienting an object with three robots. The top line denotes three snapshots of the first step. The second line denotes three snapshots of the second step. The black circles denote stationary robots. The white circles denote active robots.

A planner [Rus] that takes as input the geometric model of the object we wish to manipulate. It outputs the initial contact locations for the robots and the order in which the robots become active, which is called the push-tracking *schedule*. Moreover, the termination of each each step is characterized by "jamming" the object between the three robots and thus can be detected by an active slip sensor. The planner guarantees that *no robot will break contact* in a multi-step push-tracking reorientation protocol.

The output of the planner is used for the *setup* phase, in which the robots make contact with the object in the areas generated by the planner, by using motion plans generated by a trajectory planner. The output of the planner is also used by the global controller to signal the robots when to become active.

The planning system diagram and the execution system diagram are depicted in Figures 3(a), and 4(a).

A system of robots executing this algorithm requires the following *skills* of each robot:

- `(goto` $\langle$*position*$\rangle$`)` to contact the object during the setup phase in the areas computed by the planner. `(goto` $\langle$*position*$\rangle$`)` can be implemented by using a trajectory generator and a localization system, or even dead-reckoning [Bro].
- `(push-track)` to control the pushing-tracking motion of each active robot. This entails exerting a force in the normal direction to the face of the object while commanding a velocity in the tangential direction (see Figures 2 and 1.) This can be implemented with a hybrid controller [RC] which servoes on the orientation of a face of the manipulated object.
- `(active-slip?)` used by the stationary robots to detect slip at their points of contact. The value of this predicate is passed to the global controller which evaluates the termination predicate.

2.2 An Off-line, Local Control Protocol

A variant of this protocol denoted by LOCAL-OFFLINE can be derived for three (or more) independent robots. This system of autonomous robots cooperates to complete a reorientation does not have a global controller. Instead, the robots use communication (line of sight IR or RF) to synchronize their actions.

Three robots suffice in this protocol. At every point in time, there is one active robot and two stationary robots. Under the assumption that the robots begin in contact with the object and each robot knows the push-tracking schedule, the algorithm can be summarized as follows:

- Active robot:
  1. push-track until position or force termination
  2. signal the next robot in the schedule
  3. become stationary
- Stationary robots:
  1. sense for relative motion (*slip*)
  2. if no slip, signal the active robot
  3. when signalled, become active

The same planner as in the algorithm GLOBAL-OFFLINE (Section 2.1) is used to plan the setup phase and to determine the push-tracking schedule (see Figure 3(b)). The schedule is used for communication and it can be represented as a directed graph. The nodes of this graph are robots and the edges are communication paths as in Figure 4(b). This graph is a directed Hamiltonian cycle. The planner can be thought of as giving the topology of the communication graph. A token-passing mechanism can be used to achieve the necessary communication. Note that in this scheme, the active robot receives messages from one robot only. Since all the stationary robots can sense the "no slip" condition approximately simultaneously, the active robot will receive the signal.

Consider the skills necessary for a team of robots to perform this algorithm. In addition to the skills of GLOBAL-OFFLINE, the robots need the ability to communicate turn-taking conditions to each other. Another difference is the implementation of push-tracking and of the sensor for the jamming condition. The key issue is that an autonomous mobile robot rarely has the sophistication required for hybrid control or for active slip sensing. Our wheeled robots are moving kinematic constraints (from a control point of view.) We implement the push-tracking by choosing a suitable velocity direction and then servo on the relative angle between the robot and the object, which our robots can measure directly [JR]. Thus, a new skill is required of each robot, namely the ability to measure the relative orientation at a point of contact. We implemented the sensor for the termination of a pushing-tracking step (the "jamming" condition) by allowing the robots to detect impediment conditions thresholded by force [RD, Jen]. In summary, the following skills are required of each robot involved in this protocol.

- `(goto` ⟨*position*⟩`)` and `(push-track)`
- `(align)` Each robot has a relative orientation sensor that allows it to align itself to a specified angle to the face of the object. Aligning is executed in parallel with push-track and is described in Figure 6.
- `(blocked?)` The active robots must be able to detect the "jamming" condition. This is implemented in our system by sensing *impediment* conditions.
- `(pass-token)` Active robots have the ability to pass the pushing-tracking control to stationary robots.
- `(receive-token)` Stationary robots have the ability to receive a signal that makes them active.

2.3 An On-line, Synchronous Protocol

The two previous protocols require a planner. We now ask: do reorientation protocols depend crucially on planning? We present another version of the algorithm (which we denote by SYNCH-ONLINE) that does not use a geometric model for the object being manipulated and does not necessitate a planner.

Three robots suffice for this algorithm. At any point in time there is one active robot and two stationary robots. Under the assumption that the robots begin in contact with the object, we have:

- Active robot:
  1. push-track until sensing contact break or impediment
  2. if contact break than *spin*
  3. else if impediment then signal a stationary robot and become stationary
- Stationary robots:
  1. wait for a signal
  2. if contact break than guarded-move to contact
  3. when token received, become active

In this algorithm, *spin* is a motion that is executed when an active robot has gone past the end of the face. In this case, the robot turns around and reacquires contact by traveling around a circular trajectory (see Figure 1(b).)

Let us determine precisely what is the information provided by the planner and consider removing it. Without a reorientation planner:

1. There are no initial contact regions for the robots. However, in the setup phase, the robots can acquire contact by executing guarded moves in directions that are random but towards the center of the object. The robots can distribute themselves randomly around a circle about the object in the same manner as any formation algorithm. (For example, we can use a potential field with a large repeller at any point inside the object and small repellers on each robot.) Hence we must add randomization, but only in the setup phase.

and 2. There is no particular sequencing of push-tracking steps (as opposed to having a fixed sequence computed by the planner), thus there is no particular communication graph for the reorientation phase. As a result, there can be no guarantee that the robots maintain contact at all times. While maintaining contact is important for fine manipulation within a dexterous hand, it is not necessary for the task of large-scale manipulation with mobile robots. Therefore, we are willing to give up maintaining contact in favor of an online algorithm without a planner and without a world model. We cope with robots breaking contact in the following way. When a stationary robot breaks contact it executes a `(guarded-move)` (forward) to reacquire contact. When an active robot breaks contact, it has effectively gone past the end of a face. This condition, which we label as `(end-of-face?)` (see Figure 2) can be monitored by using a robot's contact bumpers. When a robot goes past the end of a face, it reacquires contact by `spin` (see Figure 2). In `(spin)` the rate and radius of rotation can be chosen so that in our scale and speed regime the robot is guaranteed to reacquire contact.

An additional problem may occur. Without a planner's Hamiltonian cycle the agents may immobilize the object in a configuration for which no pushing action by any agent will result in motion. This condition must be detected. When it is detected, the robots randomize to pick a new configuration around the object. To randomize, every robot executes a `(spin)`.

Figure 3(c) describes the planning system and Figure 4(c) describes the communication graph for this protocol. The following skills are required of each robot involved in this protocol.

- `(push-track)`, `(align)`, `(pass-token` ⟨*token*⟩`)`, `(receive-token)`, and `(blocked?)`

- `(guarded-move` $\langle direction \rangle$`)` to reacquire contact
- `(end-of-face?)` The active robot must be able to detect when sliding past the end of the face.
- `(spin)` Each robot must be able to reacquire contact when the end fo the face is reached.

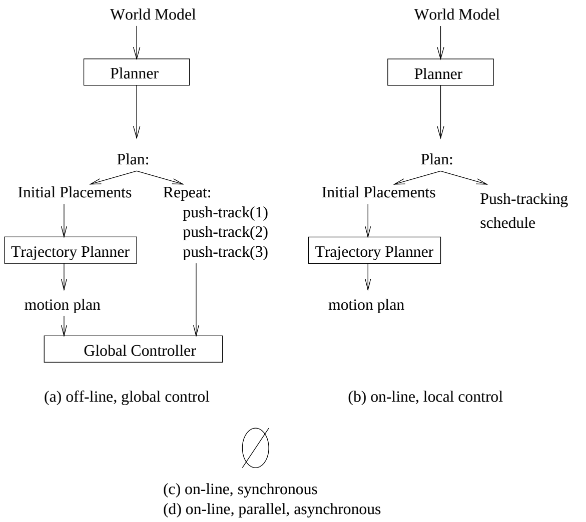

Figure 3: Planning System Diagram. Data flow, information processing, use of world models, and output of the planners. Arrows show I/O structure (data input and output) to and from computational processes. Note that Planner (b) can be obtained from Planner (a) by filtering the push-tracking *schedule*.

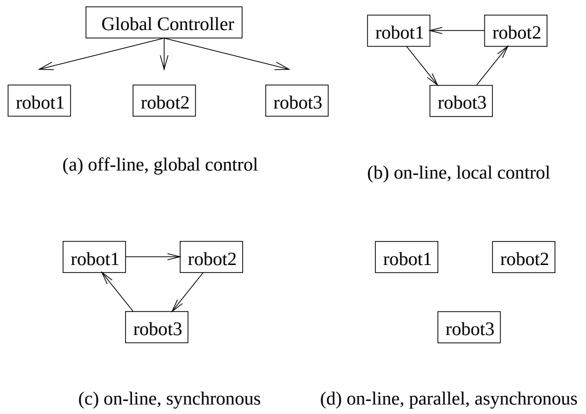

Figure 4: Execution System Diagram and Communication Graph. Arrows show messages passed using *direct* communication. Note that the communication graph in (b) is determined by the pushing schedule, while the communication graph in (c) is any randomly selected Hamiltonian cycle.

2.4 An On-line, Uniform, Asynchronous Protocol

Figure 5: Two mobile robots cooperating to reorient a couch: a snapshot taken from a couch reorientation experiment.

The three previous protocols require explicit communication. We now ask: how essential is the explicit communication for reorientation protocols? We present a protocol (which we denote by ASYNCH-ONLINE) that is *uniform* (SPMD), in that the robots execute the same program asynchronously and in parallel and there is no explicit communication between the robots.

For this protocol, two robots suffice. All the robots are active all the time. Assuming that the robots are in contact with the object, the following algorithm achieves the reorientation.

- Active Robots:
  1. push-track until contact break or impediment
  2. if contact break then spin
  3. if impediment then graze

The intuition is that the robots try to maintain the push-tracking state. When the end of the face is reached, the *spinning* motion is employed to reacquire contact. When an impediment is encountered, the robot executes a guarded move near-parallel to, but towards the face of the object, effectively *grazing* the object. Graze terminates when the robot recontacts the object or when it detects that it has traveled past the end of the face. Hence, graze terminates in (I) reacquiring the object at a contact with a longer moment arm, or (II) missing altogether. (I) is detected with a guarded move. (II) is detected using a sonar-based wall-corner detection algorithm of [JR]. When (II) occurs, the robot executes a `(spin)`.

In the setup phase the robots contact the object using the same procedure as in protocol SYNCH-ONLINE. The robots may reach the object at different times. As soon as a robot makes contact with the object, it proceeds with executing a push-tracking motion.

Figures 3(d) and 4(d) describe the robot activities in this protocol. Note that there is no explicit communication. The following skills are required of the robots.

- `(guarded-move` $\langle direction \rangle$`)`, `(push-track)`, `(align)`, `(blocked?)`, `(end-of-face?)`, and `(spin)`
- `(graze)` Each robot must be able to translate near-parallel to the face of an object to find a new contacting point. Graze uses sonar.

We have implemented a system to move furniture in our lab that includes the asynchronous online reorientation protocol. Our experiments suggest that the system is very robust.

3 Analyzing the protocols

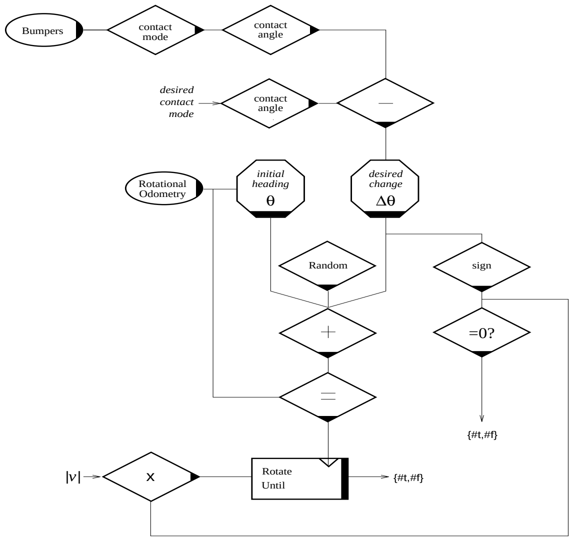

Figure 6: Circuit for `(align)`. The robot has a ring of contact bumpers and the goal is to position the robot so that the two front bumpers (for example) are compressed. The robot reads its bumper configuration and either terminates, rotates clockwise, rotates counterclockwise, or randomly repositions itself on the face of the object depending on the bumper values. If the two front bumpers are compressed it terminates. If a subset of the right bumpers are compressed it rotates counterclockwise. If a subset of the left bumpers are compressed it rotates clockwise. If the robot fails to align after several rotates, it reacquires contact by randomization.

We now ask, how do we analyze our manipulation protocols? The idea is to carefully count the resources and to model the protocols as circuits [DJR, Don]. Protocol comparison can then be modeled as transformations between the circuits. [Don] describes how to model sensori-computational systems as circuits. The circuits are then modeled as graphs. Vertices correspond to different sensori-computational components of the system (resources) and edges correspond to data paths through which information passes. Vertices can also correspond to actuators in which case we obtain *manipulation circuits* ([Don], pg 250). We define the operator + to combine two circuits and the operator * to *permute* a circuit. Here is what *permute* means: given circuits S and Q, suppose it is possible to redistribute the resources on Q to build a circuit with the functionality of S without consuming or adding new resources. This redistribution is called a *permutation* [Don] and it is denoted by Q^*.

Each one of the skills required by a reorientation protocol can be modeled as a circuit. While the entire circuit for a protocol is too detailed to show here, we illustrate their flavor by an example: the circuit for `(align)` (see Figure 6 and [JR] for a detailed explanation and analysis).

Briefly, the circuit is a pictorial representations of robot programs. These programs, while expressed easily and succinctly in SCHEME, are shown in circuit form to highlight their resource usage. Specifically, we want to track the use of: *sensors* — shown as ellipses; *computation* — shown in diamonds; *control of effectors* — rectangular boxes; and *state* — the octagons, which function as latches.

It is important to note two things: (1) The circuits are direct translations of programs; and (2) they are not flowcharts, but are more analogous to electrical circuits or "circuits" in the theory of computation.

3.1 Reductions

We now review some definitions from [Don]. *Calibration* exploits external state. *Calibration complexity*, 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.

A *manipulation circuit* is a sensory-computational circuit containing actuators. An actuator is a component whose output forces the configuration of the system to change or evolve through a dynamics equation. For example, figure 6 is a manipulation circuit.

Let S and Q be manipulation circuits. We say S *weakly simulates* Q when they have identical forward-attracting compact limit sets in state space. The *maximum bandwidth* of a manipulation circuit S is the largest-capacity data path in S, or the output size of S, whichever is greater. We now define a special but primitive "circuit" whose only purpose is communication. This circuit is called COMM(S) and contains a data path whose bandwidth is as large as the maximum bandwidth of S.

Definition 3.1 *We write $S \leq Q$ when (1) Q weakly 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.*

Definition 3.2 *We write $S \leq_0 Q$ if there exists a (graph) permutation Q^* of manipulation circuit Q such that $S \leq Q^*$.*

Definition 3.3 *Consider two manipulation circuits S and Q. We say S is* efficiently reducible *to Q if*

$$S \leq_0 Q + \textsc{comm}(S). \tag{1}$$

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

When equation (1) holds, we know that there exists a permutation $(Q + \textsc{comm}(S))^*$ of the combined circuit $Q+\textsc{comm}(S)$, such that $S \leq (Q+\textsc{comm}(S))^*$. Specifically, the manipulation circuit $(Q+\textsc{comm}(S))^*$ may be implemented as the circuit Q permuted in an arbitrary way, plus one extra data path whose bandwidth is that of the largest flow in S.

The reduction $\leq_1$ (Definition 3.3) is a "1-wire" reduction. It does not appear to be transitive. The reduction $\leq_0$ (Definition 3.2) is a *"0-wire"* reduction. It is transitive for a particular class of circuits called *simple systems* in [Don]. Simple systems can be defined using only algebraic equalities (*i.e.*, general algebraic relations are not required.) We could analogously define a *k-wire reduction* $\leq_k$ by modifying the equation in definition 3.3 to read $S \leq_0 Q + k \cdot \textsc{comm}(S)$, where $k \cdot \textsc{comm}(S)$ denotes $\overbrace{\text{comm}(S) + \cdots + \text{comm}(S)}^{k \text{ times}}$.

When $S \leq_k Q$ and $Q \leq_k S$ we write $S =_k Q$. When $S \leq Q$ and $Q \leq S$ we write $S \equiv Q$. Both are equivalence relations. In general, the combined circuit $S+Q$ denotes the graph union of S and Q—this makes sense since S and Q are modeled as embedded graphs. We will only consider logical sentences of the form "there exists a permutation $(S+Q)^*$ of the combined sensor system $S+Q$ ..." and hence we do not need to worry here about how S and Q are combined. (The existentially quantified permutation "guesses" the correct combination.) Finally, if Q is a subset of S we denote by $S-Q$ the circuit obtained by removing all elements of Q from S.

3.2 Tradeoffs and Equivalences

Consider the reduction $\leq_0$, and suppose that $S \leq_0 Q$. If S reduces to Q then there exists a permutation Q^* of Q such that $S \leq Q^*$. Note that $S \leq Q$ implies that $S \leq_0 Q$. This is because the identity permutation (no change) of Q is a valid (if trivial) permutation of Q. Hence, if $S \leq Q$, we take $Q^* = Q$ to obtain $S \leq_0 Q$.

Let A and B be any circuits. In this section we will think of A and B as "resources"—for example, a primitive action or sensing operation. Suppose that we know

$$S =_0 Q, \tag{2}$$

and we also have

$$S =_0 (Q - B + A). \tag{3}$$

Equation (3) is shorthand for $S \leq_0 (Q - B + A)$ and $Q \leq_0 S - A + B$.

Equations (2) and (3) together argue that $B - A = 0$, in other words, that B and A trade-off, or are equivalent. One caveat is that "$B = A$" is only valid for the task specified in the codesignation constraints [Don] of Q and S. For example, we prove resource trade-offs for our four manipulation circuits; but we only claim these trade-offs for the reorientation task. For some other task, resources A and B may not function equivalently[5].

This gives the following procedure to prove trade-offs for manipulation circuits X and Y and resources A and B. The key idea in the following is that the relation $=_0$ in equations (2) and (3) can be different—for example, $\equiv$, $=_m$, or $\leq_k$. Even in these cases the resulting information invariant equations provide us with a means of formalizing and proving trade-offs: the catch is that the strength of the trade-off is determined by the strength of the relation.

To prove a trade-off between A and B in the information invariant framework:

1. First establish that $X R_n Y$, where R_n is a relation in the set $\{\equiv, =_n, \leq_n\}$ and n is an integer. Note that the n is not needed for $\equiv$.

2. Then show that $X S_m (Y - A + B)$ where $S_m \in \{\equiv, =_m, \leq_m\}$, m is an integer and $m \geq n$.

3. The information invariants formalism then argues that A and B are equivalent or trade-off. More precisely, $(B - A)T0$, or BTA, where the "strength" of the relation T is defined by the strength of R_n and S_m. That is, $\equiv$ is stronger than $=$ which is stronger than $\leq$, and smaller subscripts are stronger than larger subscripts. This defines a partial order on the space of reduction types $(\{\equiv\} \cup (\{=, \leq\} \times \mathbb{N}))^2$, and this partial order in turn defines a hierarchy of trade-offs.

This argument only constitutes a proof in the case where systems X and Y are *minimal* systems (a system X is *minimal* for a task if there is no resource A that is a subset of that system such that the system $X - A$ can accomplish the task.) We can also obtain a proof in the case resources A and B are indispensible resources even when systems X and Y are not minimal. At present, we can only argue empirically that our protocols are near-minimal. Until a proof of the minimality is obtained, we challenge the reader to design and test manipulation protocols with fewer resources!

We now apply this notion of tradeoffs to our four protocols. The following theorem summarizes the results in terms of the circuits that correspond to the robot skills:

Theorem 1 *The following reductions and tradeoffs hold:*

1. GLOBAL-OFFLINE $\leq_0$ LOCAL-OFFLINE + *global-control* + 3 `(active-slip?)` − 3 `(blocked?)` − 3 `(align)`

2. SYNCH-ONLINE $=_0$ LOCAL-OFFLINE − *planner* − 3 `(goto)` + 3 `(guarded-move)` + 3 `(end-of-face?)` + 3 `(spin)`

[5] To prove trade-offs across a class of tasks, we must use a mechanism called "universal reduction" [Don], which is beyond the scope of this paper.

3. ASYNCH-ONLINE $\leq_{-3}$ SYNCH-ONLINE + 2 (graze) − (end-of-face?) − (spin) − (guarded-move) − (align) − (push-track) − (blocked?)

4. SYNCH-ONLINE $\leq_3$ ASYNCH-ONLINE − 2 (graze) + (end-of-face?) + (spin) + (guarded-move) + (align) + (push-track) + (blocked?)

Note that in part 3 (4) of the theorem, the circuits are subtracted (added) because ASYNCH-ONLINE uses two robots while SYNCH-ONLINE uses three.

Since all four protocols are equivalent (Section 2) as dynamical systems, for any pair of protocols X and Y we have $X \equiv Y$. Since each clause of Theorem 1 has the form $X \leq_k Y - B + A$, each clause argues that resources A and B trade off with strength $(\equiv, \leq_k)$.

4 Discussion

The transformational method discussed here can be thought of as a "minimalist" approach to manipulation. In the same spirit as [EM] who ask "can a robot do manipulation without sensing?", this work investigates "what manipulative tasks can robots do without global control, communication, planning, or synchronization?" We have shown how this methodology introduced in [DJR] can be used to develop and analyze parallel, on-line reorientation protocols.

Acknowledgements

This paper describes research done in the Robotics and Vision Laboratory at Cornell University. Support for our robotics research was provided in part by the National Science Foundation under grants No. IRI-8802390, IRI-9000532, IRI-9201699, and by a Presidential Young Investigator award to Bruce Donald, and in part by the Air Force Office of Sponsored Research, the Mathematical Sciences Institute, Intel Corporation, and AT&T Bell laboratories.

References

[BR] A. Back and D. Rus, Towards task-directed coordinated manipulation, in *Proceedings of the IEEE/RSJ 1993 International Conference on Intelligent Robots and Systems*, Yokohama, Japan, 1993.

[BHD] R. Beckers, O. Holland, and J. Deneubourg, From local actions to global tasks: stigmergy and collective robotics, in *Artificial Life IV*, R. Brooks and P. Maes eds., MIT Press, 1994.

[BBDJR] K. Böhringer, R. Brown, B. Donald, J. Jennings, and D. Rus, Distributed Robotic Manipulation: Experiments in Minimalism, in Proceedings of the International Symposium on Experimental Robotics, Stanford, CA, 1995.

[Bro] R. Brown, *Localization, Mapmaking, and Distributed Manipulation with Flexible, Robust Mobile Robots*, PhD thesis, Department of Computer Science, Cornell University, available as technical report TR95-1485, 1995.

[CFKM] Y. Cao, A. Fukunaga, A. Kahng, and F. Meng, Cooperative Mobile Robots: Antecedents and Directions, Technical Report, UCLA Department of Computer Science, 1995.

[Don] B. Donald, Information Invariants in Robotics, *Artificial Intelligence* (72), pg. 217-304.

[DJR] B. Donald, J. Jennings, and D. Rus, Information invariants for distributed manipulation, in *Procs of ISRR 1993.* Also in *The First Workshop on the Algorithmic Foundations of Robotics*, A. K. Peters, Boston, MA ed. K. Goldberg, R.Wilson and J.-C. Latombe (1994).

[EM] M. Erdmann and M. Mason, An exploration of sensorless manipulation, *The IEEE Journal of Robotics and Automation*, 4(4):369-379, 1988.

[Erd2] M. Erdmann, Understanding Action and Sensing by Designing Action-Based Sensors *International Journal of Robotics Research,* in press.

[Jen] J. Jennings, *Distributed Manipulation with Mobile Robots*, forthcoming PhD thesis, Department of Computer Science, Cornell University (forthcoming, August 1995.)

[JR] J. Jennings and D. Rus, Active model acquisition for near-sensorless manipulation with mobile robots, in *Proceedings of the IASTED International Conference on Robotics and Manufacturing*, 1993.

[Mas1] M. T. Mason, Manipulator grasping and pushing operations, *International Journal of Robotics Research*, 5(3):53-71, 1986.

[OY] Y. Okawa, and K. Yokoyama, Control of a Mobile Robot for the Push-a-Box Operation, in *Proceedings of the IEEE ICRA*, Nice, France, 1992.

[Par] L. Parker, *Heterogeneous Multi-Robot Cooperation*, PhD thesis, MIT EECS department, 1994.

[RC] M. H. Raibert and J. J. Craig, Hybrid position/force control of manipulators, *Journal of Dynamic Systems, Measurement and Control*, vol.102, 1981.

[RD] J. Rees and B. Donald, Program Mobile Robots in Scheme, in *Proceedings of the IEEE International Conference on Robotics and Automation*, Nice, France, 1992.

[Rus] D. Rus, *Fine motion planning for dexterous manipulation*, PhD thesis available as CU-CS-TR 92-1323, Department of Computer Science, Cornell University, 1992.