1-1-1995

Vision and Force Driven Sensorimotor Primitives for Robotic Assembly Skills

J. Dan Morrow
Carnegie Mellon University

Bradley J. Nelson
Carnegie Mellon University

Pradeep Khosla
Carnegie Mellon University, pkhosla@cmu.edu

In the Proceedings of the 1995 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Pittsburgh, PA August 5-9, 1995, Vol. 3, pp. 234-240

Vision and Force Driven Sensorimotor Primitives for Robotic Assembly Skills

J. Daniel Morrow† Bradley J. Nelson† Pradeep K. Khosla‡

†Robotics Ph.D. Program
‡Dept. of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Integrating sensors into robot systems is an important step towards increasing the flexibility of robotic manufacturing systems. Current sensor integration is largely task-specific which hinders flexibility. We are developing a sensorimotor command layer that encapsulates useful combinations of sensing and action which can be applied to many tasks within a domain. The sensorimotor commands provide a higher-level in which to terminate task strategy plans, which eases the development of sensor-driven robot programs. This paper reports on the development of both force and vision driven commands which are successfully applied to two different connector insertion experiments.

1 Introduction

Creating sensor-based robot programs continues to be a formidable challenge. Two contributing factors are programming difficulty and lack of sensor integration. Addressing these problems simultaneously is important because they are coupled: introducing sensors exacerbates the programming problem by increasing its complexity. We propose the development of a sensorimotor layer which bridges the robot and sensor spaces with the task space. In this paper, we introduce the ideas behind sensorimotor primitive (SMP) development and provide examples of both force and vision driven SMP's. In addition, some of these SMP's are implemented and used to construct sensor-based control strategies for executing two different connector insertions (D and BNC).

Our goal is to build a richer set of command primitives which effectively integrate sensing into the command set for a particular class of tasks (e.g. rigid-body assembly). The goal is to provide the task programmer with higher-level commands which incorporate sensing and are relevant to the task domain. The critical benefits of sensor integration are 1) hiding low-level details of processing sensor information, and 2) embedding generic task domain knowledge into the command set. The first goal is achieved by creating small, reconfigurable modules which process

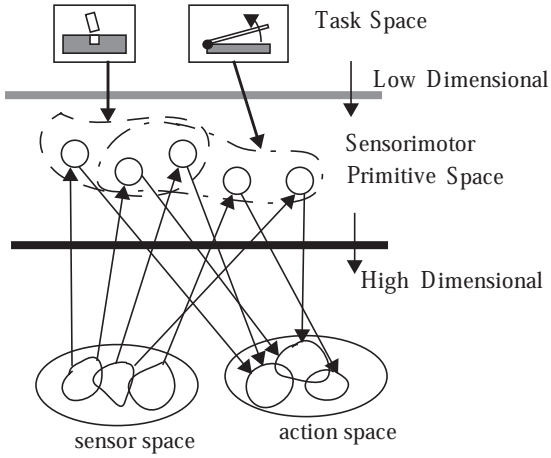


Figure 1: Sensorimotor Space

and use sensor information; this allows us to leverage the application of a sensor since it is encapsulated. The second goal is achieved when the models used to interpret sensor information are applicable to many tasks within a domain. Whenever a sensorimotor primitive is developed, some knowledge about the task (and about the domain if the knowledge is sufficiently general) is encapsulated as well. The problem is identifying common models for sensor interpretation which apply to a variety of related tasks. To the extent that these models are applicable only to very similar tasks, the task domains will be exceedingly small for which the command set is applicable. The challenge is to construct a sensor-integrated command set with enough embedded knowledge to reduce the difficulty of the programming task while retaining enough generality to have wide applicability.

1.1 Related Work

Many researchers [16][19] refer to skill libraries or task-achieving behaviors as a source of robust, skill-achieving programs. This postpones (but does not remove) the very difficult issue of how to synthesize such skill libraries. The sensorimotor layer is a direct effort to ease the programming of robust, sensor-based skills.

Other researchers have suggested robot control primitives. Lyons [10] has developed a theory of computation for sensor-based robotics; his *robot schema* computational element is very similar to our Chimera reconfigurable module [17]. Brockett [1] suggests a postscript-type programming language for robotics in which task strategies can be described independently of a particular robot system. Deno et al [3] discuss control primitives which are inspired by the hierarchical nature of the neuromuscular system, but these do not have a strong connection to the task. Paetsch and von Wichert [14] apply a set of heuristic behaviors in parallel to perform peg insertion with a dextrous hand. Smithers and Malcolm [16] suggest behavior-based assembly as an approach in which uncertainty is resolved at run-time and not during planning, but they do not address the issue of behavior synthesis. Most of these approaches to primitives (except [14]) are either task-specific or robot-centered. We are building on portions of this past work to make a stronger and more general connection of sensor-based control primitives to a task domain.

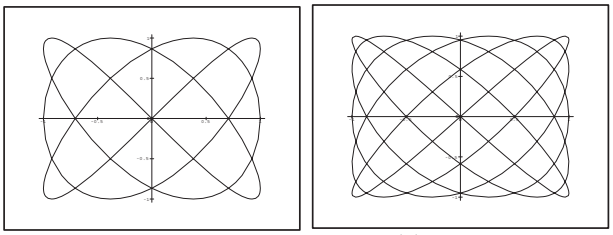
Planning robot motions based on geometric models [8] has been pursued as a method of task-level programming which reduces the programming burden. A problem with this method is that resulting strategies often fail because of inevitable errors in the task model used for planning. We believe, like Smithers and Malcolm [16], that uncertainty should be resolved at run-time, not during planning. Morris and Haynes [11] have argued that geometric models do not contain enough information about how to perform a task. They argue for using the contact constraints of the assembly task as key indicators for guiding task strategies. Similarly, we base our sensor-driven primitives on task constraints.

Much work in using force feedback has centered on detailed contact models [20]. Schimmels and Peshkin [15] have synthesized admittance matrices for particular tasks. Strip [18] has developed some general methods for peg insertion based on contact models. Donald [4] developed methods to derive plans based on error detection and recovery which are guaranteed to either succeed or recognizably fail. Erdmann [5] has investigated task information requirements through abstract sensor design. Castano and Hutchinson [6] have proposed task-based visual servoing in which virtual constraints, based on the task, are maintained. Canny and Goldberg [2] have been exploring RISC (reduced intricacy in sensing and control) robotics in which simple sensing and action elements are coupled. Many of these approaches focus on developing sensor use strategies for a particular task. We are trying to generalize sensor use for a task domain by building sensor-driven commands which are based on common task constraints in both vision and force.

2 Trajectory Primitives

Trajectory primitives are encapsulations of robot trajectory specifications. We have developed three trajectory primitives which are used in our experiments. The *movedx* primitive applies a cartesian velocity over time to achieve the specified cartesian differential motion. The *ldither* (*rdither*) primitive implements a linear (rotary) sinusoidal velocity signal at the specified frequency for the specified number of cycles. This is useful during assembly operations to locally explore.

Complex trajectories can be specified by combining trajectory primitives. For example, combining sinusoidal dithers in orthogonal directions can be used to implement an "exploration" of an area; the resulting position patterns are called *Lissajous* figures. In order to densely cover an area, the frequency ratio (n>1) between orthogonal dithers should be selected as (N+1)/N, where N is the number of cycles (of the smaller frequency sine wave) before the Lissajous pattern repeats. Figure 2 shows the Lissajous figures for two orthogonal dither signals with different values of the frequency ratio, n. Note that the positional space is well-covered by these patterns. A smaller n (closer to 1) provides more dense coverage but requires more cycles (and hence longer time) to execute.



(a) 3 cycles n=1.333 (b) 5 cycles n=1.2

Figure 2: Lissajous patterns

3 Sensorimotor Primitives

A sensorimotor primitive is a parameterized encapsulation of sensing and action which can be used to build task strategies or skills. A skill is a particular parameterized solution to a specific task (e.g. peg in hole) and is composed of sensorimotor primitives. In order to develop sensorimotor primitives, the common element(s) which relate tasks in the domain must be identified. For assembly tasks, force-driven primitives which provide for the acquisition, maintenance, and detection of different types of contact constraints are useful. Likewise, vision-driven primitives can be used to enforce positioning constraints which can be sensed on the image plane. We rely on these constraints in both the force and vision spaces to provide guidelines for sensor-driven commands.

3.1 Vision-Driven Primitives

The vision-driven primitives are based on visual servoing techniques. An image-based visual servoing approach is used, rather than a position-based approach, to avoid calculating the inverse perspective mapping of the scene at each sampling period. Thus, we must provide reference inputs to our visual servoing system in feature coordinates. To do this, desired 3D object positions must be mapped into image coordinates using a simple perspective projection model of the particular visual sensor. These primitives enforce positioning constraints on the task using poorly-calibrated camera/robot systems; errors on the image plane are used to drive the manipulator. A complete description of this visual servoing method can be found in [13].

Vision primitives are used to enforce positioning constraints on the image plane which are relevant to the task. The effective bridge between task space and robot/sensor is constructed by enforcing key task positioning constraints which can be sensed on the image plane by tracking and controlling a finite number of critical points in the task.

Image plane translation. A fundamental vision primitive is the resolution of image-plane errors through translation commands; this enforces "translation" constraints in the image plane. Castano and Hutchinson [6] have proposed a similar method to perform tasks. We use this primitive with a two-camera arrangement to enforce 3 DOF position constraints. In addition, the primitive is written so that individual combinations of axes can be controlled. This enables the flexibility needed to de-couple translation commands for certain tasks (e.g. grasping along a specific approach direction). This primitive was implemented and used in the connector insertion strategies.

Image plane rotation. Another common positioning primitive is to align lines in the image plane. Insertions, for example, can be very sensitive to errors in the insertion axis alignment. An edge-detection algorithm can robustly extract edges from an image and a primitive can use this information along with an approximate task model to align edges.

Fixed point rotation. Rotation about the normal of the image plane causes a translation of all points not on the optical axis. Therefore, one primitive involves selecting a particular point fixed with respect to the end-effector which is relevant to the task and maintaining its position (in the image) during a rotation.

Visual grasping. One primitive which can be very useful is a vision-guided primitive from a camera mounted on the gripper -- so-called "eye-in-hand" primitives. This can be used to align the gripper with cross-sections which are extracted from binary vision images. Automatic centering and obstacle avoidance could be implemented with such a primitive.

3.2 Force-Driven Primitives

Guarded move. This primitive is the common guarded move in which straight-line motion is terminated by contact. The contact is detected by a force threshold in the direction of motion. The basic constraint model is a transition from free space (and complete motion freedom) to a point/plane contact where one direction DOF has been removed.

Sticking move. The "stick" primitive involves the transition, upon contact, to a "maintenance" velocity which will maintain contact with a surface when used with a damping controller. In addition, the cartesian position is monitored in the direction of the maintenance velocity, and if the displacement exceeds a specified threshold, the primitive detects this and terminates. This prevents the robot from continuing to move when contact has been lost, and encapsulates the maintenance of point/plane contact with loss detection.

Accommodation. The sub-matrices of a 6x6 damping matrix which provides accommodation control can be viewed as sensorimotor primitives. The most common one is linear accommodation: complying to linear forces by performing translations. A sensorimotor primitive which introduces angular accommodation in response to torques and forces implements a remote-center-of-compliance [20] useful for peg insertion tasks.

Correlation. Active sensing primitives, which use the commanded action to process the sensor signal, are effective ways of extracting information from biased and noisy sensor signals [7]. We have employed a correlation technique to detect when the reference command is perturbed by the damping controller, indicating the presence of a motion constraint. The correlation (C) is computed with the following equation:

$$C = \frac{\left(\sum_{i=0}^{N} f\left(\frac{i2\pi}{N}\right) g\left(\frac{i2\pi}{N}\right)\right) N}{\sum_{i=0}^{N} \left|f\left(\frac{i2\pi}{N}\right)\right| \sum_{i=0}^{N} \left|g\left(\frac{i2\pi}{N}\right)\right|} \qquad (1)$$

For two fully correlated sinusoidal signals, the correlation value is $\pi^2/8$. Because the correlation technique is based on phase differences, the normalization is required to compensate for magnitude changes in the signals which affect the computed value.The correlation value is tested against a threshold and an event is triggered when the correlation drops below the threshold. The full development of this primitive is discussed in [12].

4 Experimental Results

Our experimental results are based on two connector insertions: a 25-pin D-shell connector and a BNC connector. Figure 3 shows diagrams of the part of each connector held by the gripper. Both connectors were stably grasped by our pneumatic, two-fingered gripper; no special fingers were constructed.

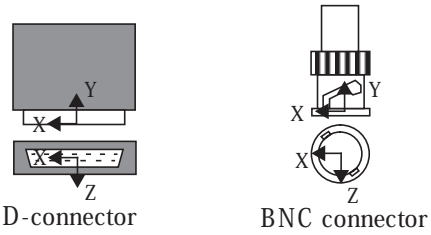


Figure 3: Connector Diagrams

The strategies are implemented as finite-state machines (FSM). Figure 4 shows an example FSM strategy in task-space which accesses the primitives in the sensorimotor space. The connector insertion strategies (Figure 5 and Figure 6) are shown in the task space with the primitives explicitly shown in the FSM. Given the small scale of the contact, we cannot reasonably derive strategies based on detailed contact-state analyses. Instead, heuristic strategies were developed based on the available command set and sensing. The strategies are based on the available command primitives (some sensor-driven, some not) and are implemented as finite-state machines (FSM). Although the different connector geometries lead to very different strategies, the same command primitives can be used to implement these strategies.

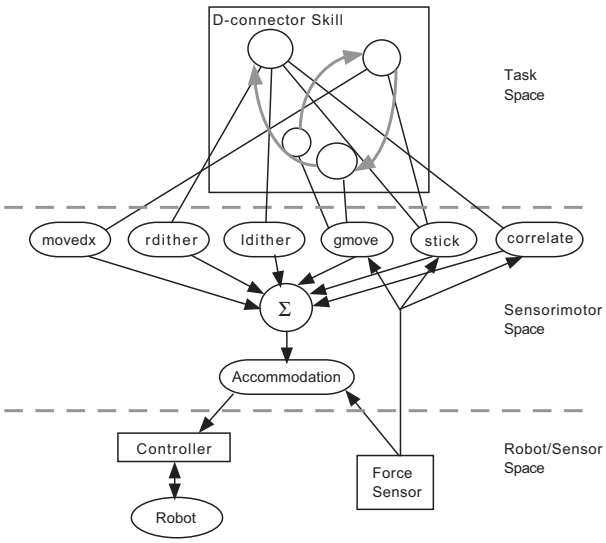


Figure 4: Finite-State Machine Strategy

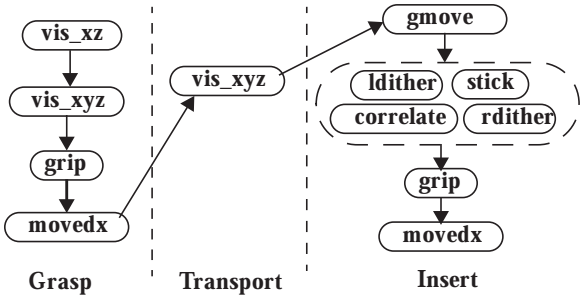


Figure 5: D-connector Insertion Strategy

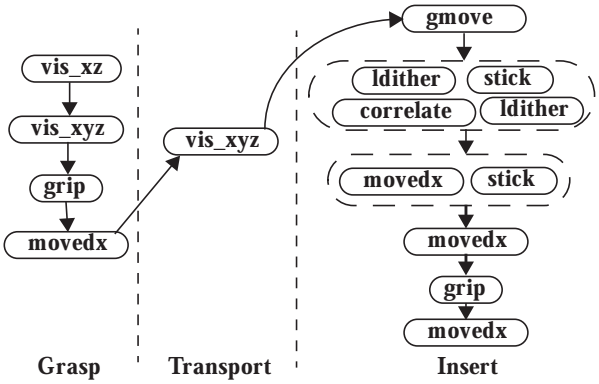


Figure 6: BNC Connector Insertion Strategy

In Figure 5 and Figure 6, each of the "bubbles" in the FSM is a Chimera module implementation of a real-time computing process. The **vis_** modules implement visual servoing primitives; for example, **vis_xz** implements visual servoing along the x and z axes. The **grip** module operates the gripper. The other modules (**gmove, stick, movedx, ldither, rdither**) are described in the force-driven or trajectory primitive sections. The task strategy implemented by primitives results in a command velocity, V_{cmd}, which is perturbed by the accommodation controller to permit contact. The perturbed velocity, V_{ref}, is used to generate joint setpoints for the robot joint controller. All of the experimental results are shown as plots of V_{ref}.

For each connector insertion task (Figure 5 and Figure 6) the strategy involves three phases: 1) grasp the connector, 2) transport to the mating connector, and 3) perform the insertion. The grasp and transport steps are dominated by vision-feedback; the insertion step is dominated by force feedback. The first step, grasping, relies on approximate angular alignment of the connector axes (X, Z) with the camera optical axes. Visual setpoints are identified in the images and controlled through visual feedback. The transport step also involves using visual feedback to position the grasped connectors above the mating connector for insertion. The insertion step is different for each task because of their different geometries; however, these two different strategies are implemented with the same set of primitives. For the D-

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.