Knowledge-Based Prehension: Capturing Human Dexterity

THEA IBERALL, JOE JACKSON, LIZ LABBE, AND RALPH ZAMPANO
School of Engineering and Science
The Hartford Graduate Center
Hartford, Conn 06120

ABSTRACT. A major question facing the development of sophisticated robotics systems is how to capture the functionality seen in versatile living systems. An approach that has proven useful in designing complex systems is to capture the explicit constraints in a knowledge based system. In this paper, we discuss a knowledge-based planning system under development that attempts to capture the versatility of human prehension. Our goal with this sytem is to both model the relationship between perceptual and motor systems in human prehension, as well as to to develop a knowledge based grasp planner able to control sophisticated, dextrous robot hands.

1. Introduction

A major question facing the development of sophisticated robotics systems is how to capture the functionality seen in versatile living systems. One such versatile function is the dexterity demonstrated by the human hand under the control of the central nervous system (CNS). Our long range goal, in modelling such a complex system, is to develop neural network algorithms that parallel the CNS approach. However, since this is beyond the current state of neuroscience, we turn to a hierarchical knowledge-based approach in order to analyze the mapping between the inputs and outputs to such a system.

As a person reaches to grasp an object, the hand shapes into a posture suitable for the interaction. In that the fingers begin shaping as soon as the arm moves, we see this as a planning problem. Some critical features about the obect have been perceived by the person that, together with the person's prior knowledge about the behavior of objects, hands, arms, and bodies, act to decide on effective prehensile behavior. In the robotics domain, various strategies have been employed for solving similar planning problems. In [15], a rule-based system is used to control the phases of problem refinement. They also identify the need for at least four separate planners in a robot system: a strategic planner, a tactical planner, a trajectory planner, and a grasp planner. In [9], a grasp planner is designed for the domain of simple assembly, which uses heuristics to prune and search a tree of all possible grasp classes relating to a particular target object. In [14], a rule-based system is used to synthesize the control for a dextrous robot hand, based on the philosophy of reflex control. Other grasp planners have been described in [10, 8]. In grasp planning for a dextrous hand, there are two issues of concern: where to grasp the object, and also with what posture to grasp it. It is for this reason that we use a two-level system for the design of our grasp planner. However, we focus more on planning the posture than on planning the location to grasp, as it is the motor problem, whereas the other is more a perceptual problem.

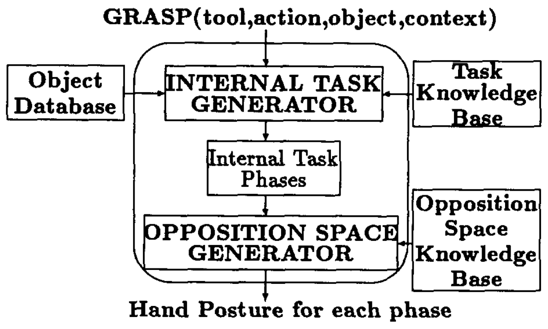

GRASP(tool,action,object,context)
Object Database
INTERNAL TASK GENERATOR
Task Knowledge Base
Internal Task Phases
OPPOSITION SPACE GENERATOR
Opposition Space Knowledge Base
Hand Posture for each phase

Figure 1: A Two-Level Grasp Planner. Tools, stored in the object database, are called up within a task context. The task is mapped into hand postures through a two-step process, based on task and opposition space constraints captured in knowledge bases.

In Figure 1, a block diagram of our grasp planner is shown. A grasp command specifies task to be performed, while pointing to an object (the one to be grasped) in an object database. Problem refinement is performed by breaking the task into phases, where each phase is a subtask described in an internal task language. These subtasks are then mapped into a series of hand postures for each phase. The external to internal task mapping and the internal task to hand posture mapping are guided by knowledge bases that contain knowledge about constraints and relationships. This use of a knowledge base is similar to the [14] approach; however, our description of hand functionality is different.

This paper describes design decisions for this system. In Section 2, we describe our high level task representation. In Section 3, we describe the object database. Our focus is generally on hand tools, as these are specialized for their

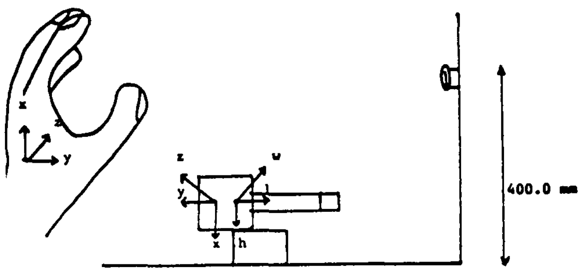


Figure 2: Example of a Screwdriver Task. Side view showing the workspace for the task. The bolt is partially driven into a post, and the screwdriver is lying on a support, making the handle accessible for grasping.

functionality. In Section 4, we describe the internal task representation. In Section 5, we describe our hand representation. In Section 6, we discuss the constraint knowledge base, focusing more on the mapping from the internal task representation to hand parameters, with less of a focus on the mapping between the external and internal task representations.

2. External Task Representation

In order to grasp an object, a grasp command (see Figure 1) is presented to the grasp planner with a set of parameters. Grasp planners in the literature have demonstrated various approaches to this problem. In [10], grasp parameters include grasp characteristics (precision, security of affixment, grasp site) and object characteristics (size, surface shape, aspect ratio, location). In [8], the important characteristics are the object's pickup location, put-down location, grasp constraints, and error bounds. The grasp command of [15] requires an indication of which feature of the object is to be grasped, along with a measure of compliance to be used during the operation, and the operation or task to be performed.

In our grasp planner, our grasp command contains pointers to a tool in the object database, an action to be performed with that tool, an object on which to perform that action, and a context for that action to occur. The task set that we are using, developed from a standard home repair manual [12], is seen in Table 1. An example of a task would be to tighten a bolt, already partially driven into a post (see Figure 2). It involves picking up a screwdriver lying on a rack on a table, and using it for the task. Our representation of this task, as it comes to the grasp planner is as follows:

```
(grasp
        (tool      'screwdriver-1)
        (action    'turn)
        (object    'bolt-6)
        (context   '(into 'board-3)))
```

This representation captures the object to be grasped (the tool) and the context within which it will be used (the action/object/context). This latter is necessary in order to help determine reasonable grasp postures needed for the task. However, the task still must be broken down into task phases, because the postures used in human prehensile behavior is potentially different for each phase, depending on the constraints acting on a given phase.

3. Object Representation

A grasp planner requires a representation which facilitates the mapping of object features to hand postures. While the objects in [9] are defined in terms of solid geometries (e.g., cylinders, etc), object features in [15] are defined in terms of surfaces (e.g., cylindrical surfaces, etc). We find this latter approach to be more useful for a grasp planner, because the hand must grasp surfaces. Vijaykumar and Arbib maintain general object characteristics (length, width, height, and weight), as well as lists of object features. Each feature has attributes, such as length, diameter, width, and a normal axis. We have added an anticipated-force-axis attribute, so that we can determine along which the expected force will be occurring in the task.

An example of the object representation for the screwdriver is seen in Figure 3. The screwdriver has eight features. Feature F1, for example, represents the round surface on the end of the handle of the screwdriver, while F2 represents the handle of the screwdriver.

4. Internal Task Representation

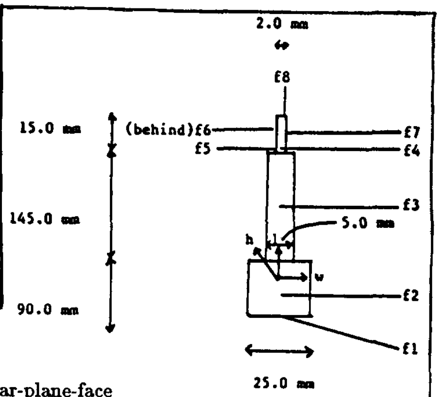
Our external task representation provides a reasonable high-level description, but it must be broken down into phases. We take this approach so that: 1) we can maintain a high-level task-level description, separate from a particular manipulator, 2) we can extract object features relevant to a particular phase from the object database, and 3) we can avoid complex issues such as determining function from shape.

In [14], an approach to grasping is presented in which they perform an alignment of major axes between the hand and the object. A more task oriented approach is taken in [2] for vision systems, where they use a task frame, a coordinate system attached to an object feature, transforming it to perform a task. We prefer this task oriented representation, thereby aligning the hand with an axis of the object within the task (a task coordinate frame) rather than with an axis of the object. We locate this task coordinate frame relative to an *opposition vector* [4,6]. All high-level task information is translated into this task frame. More than one can exist for each subtask (see [5]). An example of the internal task representation for the screwdriver is seen in Figure 4. At least two subtasks are required, one for the lifting phase and the turning phase. The information maintained in a task frame includes: opposable surface information, degrees of freedom (3 translation and 3 rotation

	GRASP COMMAND PARAMETERS			
ENGLISH EXAMPLE	ACTION	TOOL	OBJECT	CONTEXT
"drive nail into board"	drive	hammer	nail	into board
"tighten bolt into post"	turn	screwdriver	bolt	into board
"tighten nut on bolt"	turn	wrench	nut	on bolt
"extract nail from board"	pull	hammer	nail	from board
"take bolt out of post"	remove	screwdriver	bolt	from board
"take nut off bolt"	remove	wrench	nut	from bolt

Table 1: Translating task in English into parameters for the grasp command.

parameters) that the object will go through during the subtask, and the anticipated forces that will be acting on the object. The opposable surface information contains data (e.g., length, radius of curvature, etc) concerning the surfaces to be grasped. The opposition vector between these two surfaces must also be located on the object, relative to the object coordinate frame, as well as its location relative to the hand coordinate frame. Possible sources of forces and torques include object weight, opposing object force, centrifugal force, opposing object torques, and required task torques. All forces and torques are defined by a magnitude and an associated direction relative to the coordinate system of the opposition vector.



```
(screwdriver-1
 (length      250mm)
 (width       25mm)
 (height      25mm)
 (weight      .15kg)
 (features    (f1 f2 f3 f4 f5 f6 f7 f8))
(f1
    (circular-face
     (adj-features        (f2))
     (diameter            25mm)
     (normal-axis         '-l axis)
     (center              (-45mm 0 0))))
(f2
    (cylindrical-surface
     (adj-features        (f1 f3))
     (length              90mm)
     (diameter            25mm)
     (surf-normal-axis    '-w axis)
     (center              (0 -12.5mm 0))))
(f3
    (cylindrical-surface
     (adj-features        (f2 f4 f5 f6 f7))
     (length              (140mm)
     (diameter            5mm)
     (normal-axis         '-w axis)
     (center              (117mm -2.5mm 0))))
(f4
    (rectangular-plane-face
     (adj-features        (f3 f5 f7 f8))
     (length              15mm)
     (diameter            2mm)
     (normal-axis         '-h axis)
     (center              (0 -12.5mm 0))))
(f5
    (rectangular-plane-face
     (adj-features        (f3 f4 f6 f8))
     (length              15mm)
     (width               5mm)
     (normal-axis         '-w axis)
     (center              (197.5mm -1mm 0))
     (antic-force-axis    'w axis))))
(f6
    (rectangular-plane-face
     (adj-features        (f3 f5 f7 f8))
     (length              15mm)
     (width               2mm)
     (normal-axis         'h axis)
     (center              (197.5mm 0 2.5mm))))
(f7
    (rectangular-plane-face
     (adj-features        (f3 f4 f6 f8))
     (length              5mm)
     (width               2mm)
     (normal-axis         'l axis)
     (center              (197.5mm 1mm 0))
     (antic-force-axis    '-w axis))))
(f8
    (rectangular-plane-face
     (adj-features        (f4 f5 f6 f7))
     (length              5mm)
     (width               2mm)
     (normal-axis         'l axis)
     (center              (205mm 0 0))
     (antic-force-axis    '-l axis)))))
```

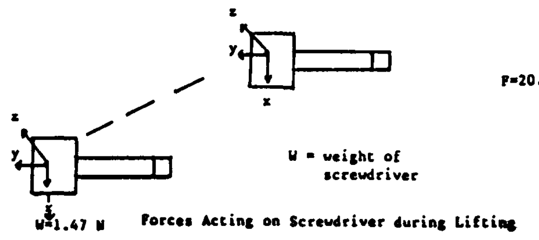
Figure 3: Example of a screwdriver object representation involves eight features (based on [15]). Attributes of these features are listed.

For lifting:

```
(opposition vector-1
                  (magnitude        25mm)
                  (loc-of-wrist     (-215mm 270mm 380mm))
                  (loc-on-obj       (0mm 0mm 0mm 90° 0 90°))
                  (opp-surface      (s1 s2))
                  (dofs             (-400mm -250mm 220mm 0 0 0))
                  (resolution       'little)
                  (forces           (1.47nt (1 0 0)))
(s1
                  (length           90mm)
                  (rad-curvature    12.5)
                  (compliance       'none)
                  (texture          'smooth))
(s2
                  (length           90mm)
                  (rad-curvature    12.5)
                  (compliance       'none)
                  (texture          'smooth)))
```

For turning:

```
(opposition vector-1
                  (magnitude        25mm)
                  (loc-of-wrist     (-30mm 110mm 0mm))
                  (loc-on-obj       (0mm -15mm 0mm 90° 0 −90°))
                  (opp-surface      (s1 s2))
                  (dofs             (0mm 0mm 0mm 0 180° 0))
                  (resolution       'little)
                  (forces           ((1.47nt (1 0 0)) (20nt (0 1 0))
                                    (1.9nt (0 -1 0))))
(s1
                  (length           90mm)
                  (rad-curvature    12.5)
                  (compliance       'none)
                  (texture          'smooth))
(s2
                  (length           90mm)
                  (rad-curvature    12.5)
                  (compliance       'none)
                  (texture          'smooth)))
```


z
y
x
z
y
W = weight of screwdriver
W=1.47 N
Forces Acting on Screwdriver during Lifting

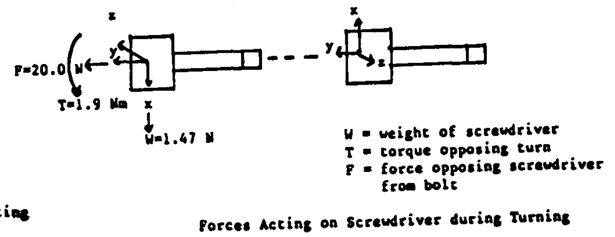

z
y
F=20.0 N
T=1.9 Nm
x
W=1.47 N
x
y
z
W = weight of screwdriver
T = torque opposing turn
F = force opposing screwdriver from bolt
Forces Acting on Screwdriver during Turning

Figure 4. Internal Task Database Entry for Screwdriver. The task is broken into two phases, one for lifting and moving the screwdriver, and one for driving the bolt into the post by turning the screwdriver.

5. Hand Representation

Once a task is refined into a set of subtasks, a hand posture is needed for each phase of the task. Several models of the human hand have been developed that categorize hand prehension in terms of symbolic postures [13,11,7,3]. Those models use symbolic hand posture names (e.g., cylindrical grasp, power grasp, and basic power) to identify specific hand grasps. In [5], we model the hand using palm, pad, and side opposition parameters. This parametric approach allows for precise definition of prehensile postures and for manipulation of object and task constraints for use in a rule base system.

The human hand is a complex device capable of applying simultaneous oppositions to an object [5]. An opposition consists of an opposing force between two virtual fingers [1,6]. Virtual finger one (VF1) consists of either the palm or the thumb. Virtual finger two (VF2) is made up of a combination of the remaining four fingers that supply the opposing force to VF1. There are certain situations that commonly require the use of a third virtual finger (VF3), although they are not considered here (see [4]).

In [5], seventy-six prehensile postures from four research papers [13,11,7,3] were represented as combinations of these three oppositions. Ignoring the redundancy, 19 unique grasps were identified, as seen in Table 2. This table shows the various ways real fingers map into virtual fingers, as well as typical combinations seen in human prehensile postures. These 19 grasps are grouped to take advantage of similarities such as the number and type of oppositions applied. While not trying to imply that there are only 7 basic grasps, these groupings instead are used to organize rule development for mapping from a task to an opposition space. As seen in Figure 5, a hand posture typically seen on tool handles involves two oppositions. The thumb, as VF1 in side opposition, presses against the side of the handle and the index finger, which is mapped into VF2. At the same time, the palm (VF1 in palm opposition) presses the body of the handle against the other fingers (VF2). A hand posture is described by the number of oppositions being used, the type of oppositions, the real to virtual finger mapping, and the length and angles of the VF vectors. Parameters that describe the functionality of the hand posture include the location of the applied force, the width of the VF surface, and the magnitude and direction of the applied force. These force related parameters are constrained by the hand posture. Once the number of oppositions, type of oppositions, and real to virtual finger mapping is performed, geometric and anatomical constraints limit the VF vector values. Since a rule base is not needed to do these algorithmic calculations, they are mentioned but temporarily ignored.

		OPPOSITION 1			OPPOSITION 2			OPPOSITION 3		
GROUP	ID	TYPE	VF1	VF2	TYPE	VF1	VF2	TYPE	VF1	VF2
A.	1.	PALM	P	I-M-R-L						
	2.	PALM	P	I-M-R						
	3.	PALM	P	I-M						
	4.	PALM	P	I						
B.	5.	PAD	T	I-M-R-L						
	6.	PAD	T	I-M-R						
	7.	PAD	T	I-M						
	8.	PAD	T	I						
C.	9.	SIDE	T	I						
D.	10.	PALM	P	I-M-R-L	SIDE	T	I			
	11.	PALM	P	I-M-R	SIDE	T	I			
	12.	PALM	P	I-M	SIDE	T	I			
	13.	PALM	P	M-R	SIDE	T	I			
	14.	PALM	P	M-R-L	SIDE	T	I			
E.	15.	PALM	P	M-R-L	PAD	T	I			
	16.	PALM	P	M-R	PAD	T	I			
	17.	PALM	P	R-L	PAD	T	I-M			
F.	18.	PAD	T	I	SIDE	T	M			
G.	19.	PALM	P	R-L	PAD	T	I	SIDE	T	M

Table 2. Nineteen Typical Postures grouped into seven groups. P=palm, T=thumb, I=index, M=middle, R=ring, L=little. Each posture identifies a combination of oppositions and a virtual to real finger mapping.

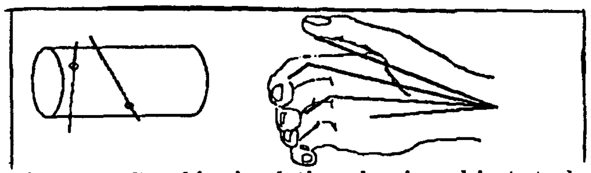


Figure 5: Graphic simulation showing object, task features, and hand parameters. A cylinder it to be lifted, using two oppositions, shown by the two opposition vectors. The two oppositions are seen in the posture taken on by the hand.

6. Opposition Space Knowledge Base

To derive hand parameters responsive to object and task representation data bases, a prototype rule base (with about 40 rules) was developed in VP-Expert, a commercially available tool. This initial rule base is not exhaustive of the domain but it does identify an approach that effectively organizes rule base development.

The first task for the rule base is to identify the possible hand grasps (see Table 2). This depends on the number of opposition vectors in the internal task representation for that phase. In the screwdriver example, there is only one opposition vector in each phase, therefore limiting our choice to hand grasps in group A, B, or C in Table 2. Next, the opposition type that will be applied to each opposition vector is determined. An example of a rule that might trigger in the example would be:

```
IF Number opposable vectors = 1 and
    Compliance = none and
    Texture = smooth and
    Radius of curvature ≥ medium and
    Force = large
THEN Opposition 1 = palm
```

The values for the parameters are symbolic (e.g,, small, medium, etc) and are designed to represent a range of values as contained in the data bases. Functions are used to convert the force representation into a subjective value (e.g., large). Once the type of opposition for each opposition vector is identified, the real finger mapping for VF1 of each opposition vector is found. An example rule in the screwdriver example would be:

```
IF Opposition 1 = palm
THEN VF1(opp1) = palm
```

Finally, the rule base is searched to identify the VF2 mappings. In this case, a VF2 mapping rule is:

```
IF Surface length ≥ 3.5 and
    Degrees of freedom = 3 and
    Compliance = none and
    Force = large and
    Force = rotational
THEN VF2(opp1) = I,M,R,L
```

where the units for 3.5 is "finger widths" and I,M,R,L means "Index, Middle, Ring, and Little" fingers.

This knowledge-base is currently being rewritten and expanded to run in Prolog on a distributed workstation.

7. Conclusions

An approach that has proven useful in designing complex systems is to capture the explicit constraints in a knowledge based system. We have shown here a system under development that attempts to capture the versatility of human prehension. An object database, containing collections of features about tools and simple shaped objects, is used as input to the grasp planner. A grasp command points to one of these objects, and provides the task context for using the object. Using two constraint knowledge base, the grasp planner first breaks the task command into a set of task requirements for phases of the task, and then determines reasonable hand postures for each of these phases. While developing this knowledge-based system, we have concurrently been developing a graphic simulator, so that the user can see a graphic representation for the hand and the object. This has been tested on an IBM-PC, and is being rewritten on a workstation in Pascal. Our long term goal, once prehensile constraints have been identified and explored, is to redo such a system in a neural network architecture. However, such a knowledge based grasp planner is useful for the control of sophisticated, dextrous robot hands designed using opposition space concepts.

8. Acknowledgements

We would like to thank G. Blessing and T. Rando for their work on the graphic simulator, D. Eldridge and T. Popierlarczyk on earlier versions of the simulator and knowledge base, and R. Vijaykumar for helping these efforts.

References

[1] Arbib, M. A., Iberall, T. and Lyons, D. (1985): Coordinated Control Programs for Movements of the Hand. In: *Hand Function and the Neocortex*, A. W. Goodwin and I. Darian-Smith (Eds), Berlin: Springer Verlag, 111-129.

[2] Ballard, D., and Hartman. (1986): Task Frames: Primitives for Sensory-Motor Coordination. *Jour Computer Vision, Graphics and Image Processing*, 36, 274-297.

[3] Cutkosky, M. R., and Wright, P. K., (1986): Modeling Manufacturing Grips and Correlations with the Design of Robot Hands, *IEEE Intl Conf Robotics and Automation*, San Fran, Calif, April 7-10: 1533-1539.

[4] Iberall, T. (1986): The Representation of Objects for Grasping. *Proc Eighth Cognitive Science Conf*, August 15-17, Amherst, Mass, 547-561.

[5] Iberall, T. (1987): The Nature of Human Prehension: Three Dextrous Hands in One. *IEEE International Conf on Robotics and Automation*, Raleigh, North Carolina, March 30- April 3, 396-401.

[6] Iberall, T., Bingham, G. and Arbib, M. A. (1986): Opposition Space as a Structuring Concept for the Analysis of Skilled Hand Movements. In: *Generation and Modulation of Action Patterns*, H. Heuer and C. Fromm, Berlin: Springer-Verlag, 158-173.

[7] Iberall, T., and Lyons, D., (1984): Towards Perceptual Robotics, *Proc 1984 IEEE Intl Conf Systems, Man, Cybernetics*, Halifax, Nova Scotia, Oct. 9-12, 147-157.

[8] Lozano-Peréz, T. and Brooks, R. (1985): An Approach to Automatic Robot Programming, *AI Memo 842*, MIT AI Lab, Cambridge, Mass.

[9] Loranzo-Peréz, T. and Winston, P.H. (1977): LAMA: A Language for Automotive Mechanical Assembly, *Proc 5th Intl Conf Artif Intel*, Cambridge, Mass, 710-716.

[10] Lyons, D. (1985): A Simple Set of Grasps for a Dextrous Hand, *COINS TR 85-37*, Dept Comp and Info Sci, Univ Mass, Amherst, Mass.

[11] Napier, J. R., (1956): The Prehensile Movements of the Human Hand, *J Bone Jt Surg*, 38B(4): 902-913.

[12] Philbin, T. (1976): *How to Make Your House Behave*, New York: Golden Press.

[13] Schlesinger, G., (1919): Der Mechanische Aufbau der Kunstlichen Glieder. In: *Ersat glieder und Arbeitshilfen*, M. Borchardt et al (eds), Berlin: Springer.

[14] Tomovic, R., Bekey, G., and Karplus, W. (1986): A Strategy for Grasp Synthesis with Multifingered Robot Hands. *Proc 1987 IEEE Intl Conf on Robotics and Automation*, Raleigh, NC, March 30-April 3, 83-89.

[15] Vijaykumar, R. and Arbib, M. A. (1986): A Task-level Robot Planner for Assembly Operations, *COINS TR 86-31*, Dept Comp Sci, Univ Mass, Amherst, Mass.