A "RISC" Paradigm for Industrial Robotics

John F. Canny*
University of California, Berkeley
Berkeley, CA 94720
jfc@ernie.berkeley.edu

Kenneth Y. Goldberg†
University of Southern California
Los Angeles, CA 90089-0273
goldberg@usc.edu

September 25, 1992

Abstract

Whitney and others have noted that it is important to make a distinction between structured and unstructured environments when designing robots. For structured environments such as factories, cost and reliability are vital to maximizing return on investment. Industrial users argue that academic researchers often neglect these criteria when developing new hardware.

In this paper we propose a paradigm for industrial robotics that we hope will appeal to both industrial users and academic researchers. We reconsider robot systems with simple sensors and actuators such as light beams and parallel-jaw grippers. Although conventional wisdom holds that such hardware lacks versatility, we review recent polynomial-time algorithms, based on this hardware, for recognizing and orienting a broad class of industrial parts.

We argue that industrial robot systems with *Reduced Intricacy in Sensing and Control* (RISC), coupled with efficient geometric algorithms, can be both inexpensive and reliable. This paradigm has a loose analogy with the recent trend toward Reduced Instruction Set Computers (RISC) in that both seek to identify a minimal set of hardware primitives and to match these primitives with highly efficient software.

1 Introduction

In the past decade, much of robotics research has focussed on anthropomorphism in the hope that once robots become sufficiently human-like, they will be able to solve all the tasks that they are currently incapable of doing. Thus there has been a quest for more degrees of freedom, hand-like configurations, tendon and direct drives, human-like visual and tactile image processing capabilities, and hardest of all, the ability to control and interpret these modules. As Whitney [24] and other have argued, while such flexibility may be necessary in unstructured environments (such as waste and hazardous materials cleanup or extra-terrestrial exploration), the cost of coordinating this flexibility in one package has been prohibitive for factory applications. While factory robots need more flexibility than they have today, they need far less than a human being.

> *"[In the 1980's, people] made robot hands that were so ungodly complex that in many cases they had no chance of standing up in a real industrial environment."* [21]

Factory users complain that excess flexibility requires excessively complex mechanisms and software, and hence costs dearly to set-up and maintain. Furthermore, reliability is often sacrificed. Certainly the market for robots has not expanded at nearly the rates predicted in the mid 1980's. A panel of industrial representatives recently came to the conclusion that "robots must be inexpensive and easy to apply. The factory floor user is not interested in, and cannot support, complex technology" [3].

In this paper we propose a paradigm for industrial robotics that we hope will appeal to both industrial users and academic researchers. Borrowing an acronym [16], we call this a "RISC" paradigm for industrial robotics, where in our case the acronym stands for **R**educed **I**ntricacy in **S**ensing and **C**ontrol. To illustrate the approach, we will explore the limits of simple sensors and actuators such as light beams and parallel-jaw grippers. While these elements are usually considered indicative of "hard automation", we argue that when used severally and together with good algorithm design, they can achieve much of the promised flexibility of anthropomorphic robots.

*Supported by a David and Lucile Packard Foundation Fellowship and by NSF Presidential Young Investigator Grant IRI-8958577

†Supported by NSF award IRI-9123747, and by a Cooperative Agreement between the National Institute for Standards and Technology and the Institute for Manufacturing and Automation Research.

One key idea of the RISC approach is to **reduce complicated operations to a combination of simpler ones** that can be performed with simple mechanisms. For example:

- Rather than using a multi-fingered hand to reorient parts, we might consider a sequence of acquire/transport/place steps with simple, grippers such as the parallel-jaw or chuck-jawed grippers specialized for insertion [12]. When more degrees of freedom are needed, pan/tilt tables might be used to achieve rotational motions. In this way we combine several devices to form a "virtual robot".

- Rather than using dense sensors such as vision to identify and locate parts, we might use a sequence of simple sensors such as light beams or potentiometers (see Sections 2 and 3 for detailed examples).

The challenge is to develop robust offline planning algorithms that will automatically perform this reduction. Although geometric motion planning with uncertainty is an active area of research [7], it is not obvious how to extend current results to systems with many degrees of freedom. By advocating simpler hardware, we hope to facilitate the application of these methods to real industrial problems. In formulating the RISC paradigm, we had the following goals in mind:

- Increased Reliability. RISC sensors and actuators have fewer components so less can go wrong. Robust offline planning algorithms will reduce online failures due to errors in sensors, control, and part models.

- Lower Start-up and Maintenance Costs. We advocate using off-the-shelf hardware whenever possible. To minimize custom programming effort, we propose offline planning algorithms (e.g to plan paths, select end effectors and choose sensor configurations). Both the offline and run-time systems must have user-friendly interfaces for the person on the shop floor.

- Increased Speed. Simple sensor data can be processed much faster. Having many simple manipulators working on task rather than one complex one allows optimizations such as pipelining and parallel execution.

As we focus on the problems of recognizing and orienting parts, which are fundamental issues in industry. In the following two sections, we describe methods that exemplify our idea of RISC sensors and manipulators. We then discuss related work and summarize.

2 RISC Sensors

RISC sensors should be simple, cheap and reliable. That is, the mechanical/electrical complexity of the sensor should be low, its information processing needs should be slight, and it should be robust. Binary optical beam sensors satisfy these criteria. In this section we show how such sensors can be used to orient polyhedral parts.

2.1 Optical Beam Sensors

Optical beam sensors are routinely used in manufacturing. But most applications only detect the presence or absence of a part or count the number of parts coming down a conveyor belt. However, when there is relative motion of the beam, these devices can provide remarkably accurate measurement of position and distance, and when several are used in unison, very accurate measurement of part orientation. They can also be used for recognition.

Since a major portion of a manufacturing system is devoted to moving the parts around, beam sensing can be performed during these motions with little impact on the sequence. Typically this is done by placing a beam sensing array on the path of the part from the feeder area to the assembly area, or on a conveyor belt that is bringing in the parts. If the part is lying on table, it can be scanned by beam sensors mounted on a gripper that is about to grasp it. This requires some additional motion of the gripper, but less than with a camera, where the camera must first be brought over the part while the gripper is out of the way, and then the camera moved away as the gripper is brought in.

The sensors we have studied consist of arrays of three beam sensors in one of two configurations. In the first configuration, the beams are crossed as shown in figure 1 and scan objects from the side. In the second configuration, the beams are parallel as shown in figure 4. We discuss each of these configurations in turn.

2.1.1 Crossed-Beam Sensor Array

Algorithms for processing crossed-beam scanner data are particularly simple. This type of scanner senses only a particular horizontal cross-section of the part, in fact it can only "see" the convex hull of this cross-section. Assuming the motion of the part is steady linear translation, the amount of time for which a beam is broken is directly proportional to the diameter of the part normal to that beam, as in figure 2. The scale factor between time and diameter is the inner product between the motion direction and the normal to the beam.

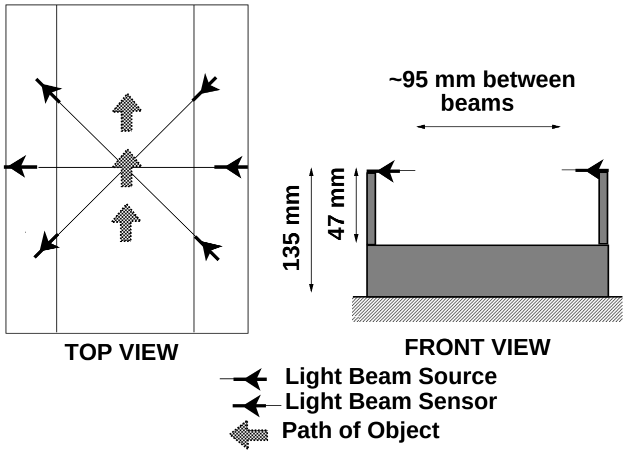

Figure 1: A crossed-beam sensor array.

With three beams, the measurements of diameter define an irregular hexagon with sides parallel to the beams. The distance between opposite sides, both of which are parallel to some beam, is the diameter normal to that beam. The sensor "sees" the smallest such hexagon that encloses the true cross-section of the part. If two different cross-sections have the same enclosing hexagon, they will be indistinguishable by the sensor. In particular, the smallest enclosing hexagon is fully determined by the convex hull of a shape. So if the cross-sections of two different parts have the same convex hull, they will not be distinguished by the 3-beam sensor, or in fact by any number of crossed beams.

If the convex hulls are different, one might be tempted to try rotating the beam sensor relative to the object, or using more beams. But some objects with different convex hull have the same diameter *at every orientation* [20]. Thus, no amount of rotation of the beams, or no increase in the number of beams, can distinguish these parts.

Such indistinguishable pairs of parts rarely occur in practice. When they do, the parallel-beam sensor can be used to discriminate them. But the crossed-beam is simpler to build, and its data is much simpler to process.

First, observe that each diameter depends only on the orientation of the part relative to a beam, not its position. For a fixed orientation of the part, we can define the part's *diameter* at direction θ to be the maximum distance between two parallel supporting lines at angle θ. Let S^1 be the set of planar orientations. The diameter function[1], $d : S^1 \rightarrow \Re$ describes how the diameter varies as the parallel lines are rotated around the part (see Figure 3). The diameter function for an n-sided polygonal part can be computed in $O(n)$ time [6].

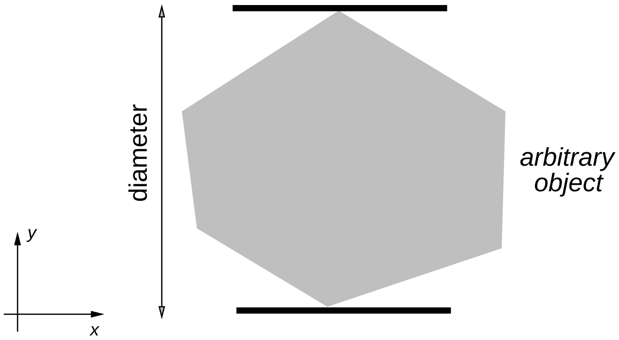

Figure 2: The diameter of a polygon

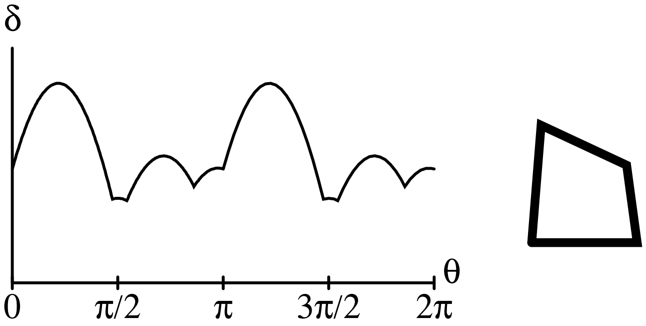

Figure 3: The diameter function for the four-sided part shown at the right in its zero orientation. During a squeeze, the part rotates so as to reduce the diameter, terminating when the diameter reaches a local minimum.

Due to noise in the measurement of diameter, a continu-

[1] Also known as the *width function* [25].

ous range of consistent part orientations are possible.

Using the crossed-beam array, all three diameter measurements depend on a single real number, the orientation of the part in the plane. Intersecting the ranges for each beam will normally produce a single (small range of) possible orientation. In fact for most orientations of a given part, two measurements suffice to disambiguate orientation. But some orientations will still be ambiguous. The third beam takes care of these *for most part shapes*. Some parts will still have a few ambiguous orientations, but these can generally be taken care of if one or two of the beams are moved (rotated).

Once the orientation of the part is found this way, it is very easy to compute its position from the absolute time of contact with the beams. So part localization for polygonal part cross-sections consists of the following steps:

1. Precompute the diameter function for the cross-section (offline).

2. Compute 3 diameters from beam-breaking times.

3. For each diameter, look up the range of possible orientations of the part relative to the corresponding beam.

4. Intersect the 3 sets of ranges to obtain a single orientation range.

5. Compute part position using the orientation information.

Each step is very simple. For more details see [23]. The diameter function can be exactly represented as an ordered list of real intervals, and a sinusoidal function within each interval. This function can be computed in $O(n)$ time, where n is the number of edges of the convex hull of the part cross-section. The same function can be used for all three beam lookups, since a rotation of a beam corresponds to a horizontal displacement of the diameter function.

The lookup of ranges of possible orientations for each diameter measurement can also be done in linear time using the representation described above. It should be clear that the number of possible orientations for each measurement is $O(n)$. Finally, intersecting three sets of ranges takes $O(n)$ time in worst case.

The constants for all these steps are very small, and since typical parts have only a few edges on their convex hull, part location can be accomplished in less than a millisecond on a state-of-the-art workstation (Sun-4 or later). The preprocessing also takes very little time, in fact it could also be done at run-time with only a slight impact on performance.

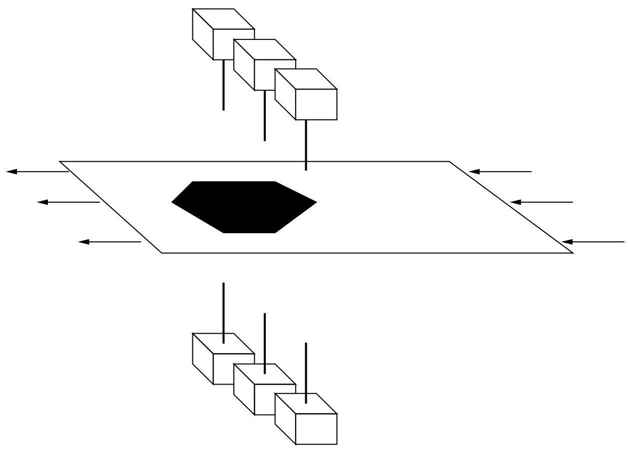

Figure 4: A parallel-beam sensor array.

2.1.2 Parallel-Beam Sensor Array

The parallel beam sensor is more powerful than the crossed beam sensor, but the algorithms for it are more complicated. It is more powerful because it can discriminate between objects that have the same convex hull (in planar section). In fact any pair of planar shapes can be discriminated, although not necessarily at all poses for a given sensor. The algorithms for processing the data are more complicated though, because the measurements do not depend only on orientation as for the crossed-beam sensor.

A sample parallel-beam sensor is shown in figure 4. It consists simply of several transmitters and receivers which process vertical beams of light that are occluded by the object. Because the beams are vertical rather than horizontal as for the crossed-beam sensor, the table or conveyor on which the part rests must either be transparent or mirrors must be used.

The parallel-beam sensor measures several linear sections of the part as shown in figure 5. For almost all polygonal parts, three beams are adequate to localize the part at all poses.

The algorithm for localizing with parallel-beam data is rather more complicated than for cross-beam data and is described more fully in [22]. We sketch it here for the convex case.

Each beam measurement defines a chord which touches the boundary of the part, as in figure 2.1.2. For any given length of chord, there are only a finite number of ways to place the chord in contact with the boundary. If the orientation of the chord is fixed, there are at most two ways to place it so it touches the boundary at both ends (for a convex polygonal object). It follows that if we know the

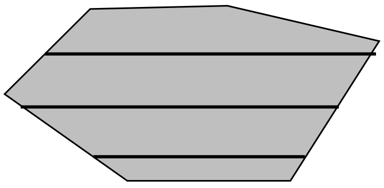

Figure 5: An example of a sensed object object; the dark lines indicate what the scanning sensors observe.

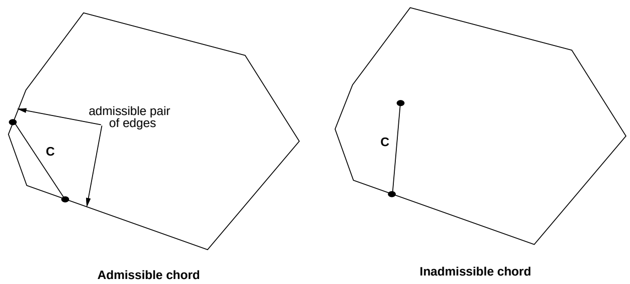

Figure 6: An admissible chord

orientation of the chord relative to the object, there will be at most two possibilities for the position of the object.

For each measured chord length, we partition the set of planar orientations into intervals such that the possible contacts (defined as the pair of edges that the chord touches) are the same. The data structure for each interval will include a list of possible edge pairs (at most two). For most convex objects with n edges, there will be n possible contact pairs. However, for pathological shapes, there may be $\Omega(n^2)$possible contact pairs for a given chord length. So this partition may contain $\Omega(n^2)$ intervals.

Within each interval, and for each of the edge pairs in that interval, the position of the object is an easily computable function of orientation. To localize the object, we compute the partitions described above for two chords. Then we merge these partitions, to create a partition which is finer than both. Within each interval of the latter partition, both possible contacts for both chords are fixed. There are at most 4 ways that *both* contacts can be specified. For each of these, we have two functions specifying the position of the object as a function of orientation. We can intersect these functions to see if there is an orientation in the interval such that both imply the same object position. If there is, then there is a valid placement of the object that agrees with both measurements. If not, we try the next interval in the partition.

This procedure will find all the possible placements of the object that agree with the measured chords. There may be more than one, although this is very unlikely. The third chord provides an additional check. The interval partition can be computed for the third chord, and then checked against each of the other two as before. While there are more direct and faster methods for checking the third chord, the method above does not increase the running time much compared to the two-chord algorithm. But it makes the *algorithm* simpler, because very little additional code needs to be written to do it. In the spirit of RISC we recommend this implementation because it makes the software simpler, hence more reliable and easier to debug.

The algorithm overall runs in linear time in the maximum number of intervals in the three partitions. While this can be as much as $\Omega(n^2)$, it is almost always $O(n)$, so this algorithm for convex objects usually takes linear time in the number of part edges n.

In the non-convex case, the situation is more complicated. The interval partitions may have at worst $O(n^2)$ intervals, the same as the convex case, but the near worst case examples are more likely to be encountered. But there is additional complexity because the number of contact pairs within each interval is no longer limited to 2, but can be as many as n. Merging two intervals requires that each feasible edge pair from one interval be checked against each feasible pair in the other, a further quadratic blow-up. But its is possible to exploit coherence to bring the complexity back down to linear in the size of the partitions. Again see [22] for details.

2.1.3 Other RISC sensors

Other examples of RISC sensors are proximity sensors, force sensors, finger gap and other kinesthetic sensors. A tactile sensor may or may not qualify, depending on its complexity and information-processing needs.

3 A RISC Manipulator

Perhaps the least complex manipulator is the parallel-jaw gripper, having one degree of freedom with binary pneumatic control (Figure 7). Although widely used in industry, conventional wisdom holds that these grippers lack versatility [14]. With a minor modification, however, these grippers can be used to recognize and orient an important class of industrial parts.

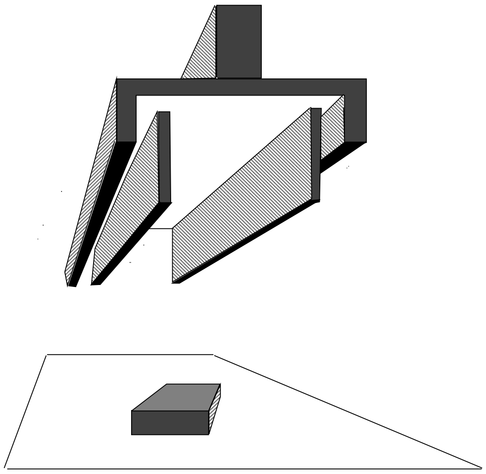

Figure 7: A parallel-jaw gripper above a rectangular part.

3.1 The Modified Gripper

The quality of a grasp configuration depends on many factors including the orientation of the part with respect to the gripper. This orientation may not be known precisely or may be disturbed by the act of grasping. For the parallel-jaw gripper grasping polygonal parts, Brost [2] defined a grasp as stable if at least three vertices of the part are in contact with the gripper jaws and any further closing of the gripper would deform the part; see Figure 8.

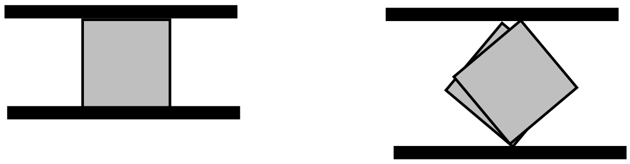

Figure 8: The grasp configuration on the left is stable; those on the right are not.

Unstable grasp configurations result from friction between the part and the jaws. This suggests that it may be desirable to eliminate friction between the object and the jaws. One approach is to coat the jaws with grease, but this has the disadvantage that the object will slip when the gripper is lifted out of the plane. We can achieve low friction in the plane of the object but high friction orthogonal to the object by mounting a sliding plate (linear bearing) on one jaw. The inner surface of both jaws is covered with a high-friction material such as rubber. See Figures 9 and 10.

To achieve a stable grasp, we close the jaws as far as possible without deforming the object. Gripper forces will cause the sliding bearing to translate until the part rotates into a stable orientation. We built a prototype of this gripper using an off-the-shelf linear bearing with a rubber band to provide spring force and a dab of grease to provide damping. We experimented with several object shapes grasped randomly in two sets of 250 trials. Without the bearing, approximately half the grasps were stable. With the bearing, every grasp was stable. Although any physical bearing experiences some friction, we conjecture that vibration arising from the gripper drive is sufficient to dislodge unstable orientations, so that the mechanism is effectively frictionless.

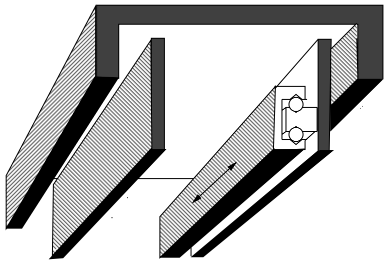

Figure 9: The modified gripper with sliding jaw. (Based on drawing by Ben Brown).

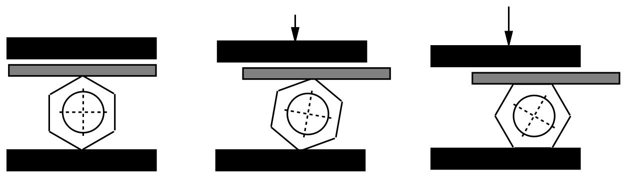

Figure 10: Time-sequence of grasping with the modified gripper. (1) As the two outer jaws close over a typical object (hex nut), horizontal forces cause the sliding jaw to translate to the left (2) until the object is gripped in a stable configuration (3). (Based on drawing by Ben Brown).

One of the primary advantages is that this modification requires no additional sensors or actuators; a low-cost and lightweight linear bearing can be easily retro-fit to any parallel-jaw gripper without requiring interface software [8].

3.2 Recognizing Parts

The parallel-jaw gripper described above can be used to recognize parts by measuring the distance between the jaws, say with a linear potentiometer. This is similar to using light beams as described in Section 2; the difference is that in this case closing the jaws causes the parts to rotate into a new configuration. For a given set of k parts with constant cross section (2.5D parts), we consider the following two problems: (1) given a set of measurements derived from random grasps of one part, decide which part was grasped. (2) find a sequence of grasp angles for the gripper, conditional on measurements, for efficiently recognizing parts from the given set.

For the first problem, since more than one part may give rise to the same diameter and the diameter sensor may be corrupted by noise due to surface compliance and backlash, we can use a Bayesian decision procedure to estimate the most probable part Since the set of grasps is random, we can assume that prior to each grasp, the part's orientation with respect to the gripper has a uniform probability distribution on the set of planar orientations. Note that each stable orientation of a given part corresponds to a minimum in the part's diameter function (see Figure 3). Thus the prior probability for each measured diameter can be derived in time $O(n)$. This becomes a conditional probability when considering a set of parts. Lacking any information to the contrary, we might assume that initially, each part is equally likely. After each measurement, the posterior probability is computed. After all measurements have been considered, we can decide on the most likely part. This method can also be adapted to allow for sensor noise using a Gaussian error model. For details see [11].

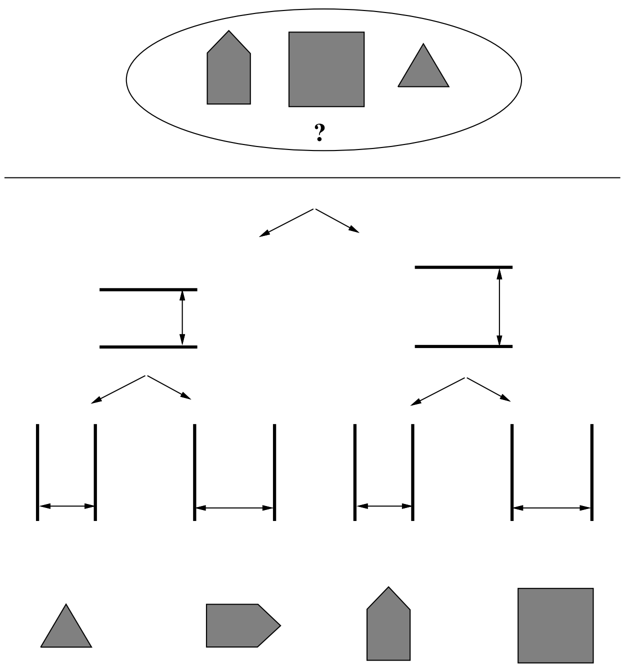

Figure 11: A grasp plan for distinguishing the three parts shown at the top.

Of course we expect to achieve better performance by tailoring the grasp strategy to the geometry of parts in the set, as shown in figure 11. This, the second problem stated above, is the planning problem. As mentioned earlier, [20] showed that some parts cannot be distinguished by measuring diameter alone. But if we restrict atte ntion to parts that are distinguishable, we can find optimal strategies by considering cliques in the following graph. Let $G = (V, E)$ be an undirected graph such that each vertex corresponds to a stable orientation from the given set of parts. Let $n = |V|$. We construct an edge between any two vertices with the same diameter. For each edge, let $R(e)$ be the set of gripper orientations that would disambiguate the neighboring vertices in a subsequent grasp ($R(e)$ is always non-empty by assumption). Let G have m edges. We can construct G in time $O(n^3)$ and it can be partitioned into disjoint maximal cliques (connected components) in time $O(n + m)$.

Each possible gripper measurement identifies one of the cliques in G. If the clique contains stable orientations from one part, we are done. If it contains stable orientations from exactly two vertices connected by edge e, we pick a gripper angle in $R(e)$, and regrasp to disambiguate between the two associated parts. However, if the clique contains more than two vertices, we look for a gripper angle in the intersection of $R(e)$ for all edges in the clique, $\bigcap_E R(e)$. If this intersection is empty, we require more than one additional grasp to identify the part. In [20] we give two planning algorithms. The first runs in time $O(n^2 \log n)$ but may not generate the shortest plan. The second finds the shortest plan but may require time exponential in n in the worst case because it considers all possible partitions of each clique. In either case, the resulting plan will never require more than n grasps. This algorithm can also be adapted to account for measurement noise.

3.3 Orienting Parts

Perhaps surprisingly, it is possible to achieve a desired final grasp configuration *without sensors*. Goldberg [6] describes an algorithm for orienting polygonal objects using the parallel-jaw gripper to grasp and ungrasp at a prespecified sequence of angles depending on object geometry. That is, for any additional stability criterion that prefers one of the stable grasp configurations over the others, we can achieve it using the compliant motion algorithm. The planning algorithm finds the shortest such sequence for any n-sided object in time $O(n^2)$. This algorithm has recently been extended to curved parts [19].

4 Previous Work

The ideas presented here are not new. In particular, Nevins and Whitney [15] stressed the distinction between structured and unstructured environments, and Whitney [24] pointed out the opposition between flexibility and efficiency in industrial machines. Many others have argued for simplicity in the factory. Mason [13] argues persuasively for replacing sensors with mechanical solutions. Others, such as Erdmann and Donald are studying ways to reduce the complexity of sensors. Our objective here is to precipitate is already in the atmosphere.

4.1 Other Examples of RISC

There are many existing examples of research that are constent with the RISC paradigm. The use of mechanical compliance in lieu of sensing was explored by, to name a few, [10, 18, 4, 5, 2, 17, 9]. Perhaps the best example is the RCC device that achieves compliant peg-in-hole assembly using passive mechanical springs rather than a complex system of force sensors and feedback control.

Another example is Scheinmann's Robot World System, which uses Cartesian arms with four d.o.f. rather than six d.o.f. serial arms. In Cartesian arms each actuator is aligned with the "natural" coordinate frame. This decouples the joints for speed and greatly simplifies programming.

The RISC paradigm shares some of the philosophy of Brooks' Insect architectures [1] in that both seek to simplify the control structure. The fundamental difference is that we intend RISC for repetitive operations in structured environments. In such cases, offline planning with explicit geometric models is essential to achieving reliable performance.

5 Discussion

Very little hardware is required to implement these examples of sensing and manipulation; light beam sensors and parallel-jaw grippers have been widely available since the 1960's. They are inexpensive and reliable. Yet these devices have been largely ignored by the research community. Conventional wisdom holds that increased complexity is required to increase flexibility. But highly articulated gripping mechanisms are not necessary for flexible manipulation. Consider chopsticks. For structured environments such as factories, we suggest that simple devices hold great promise for automation. Flexibility can be achieved by combining several simple devices as is currently done in traditional ("hard") automation. The trick is to develop new algorithms to automate the planning and set-up process. This is certainly non-trivial; as noted by the great jazz musician Thelonious Monk, "Simple ain't easy".

As examples, we have described new algorithms for sensing and manipulation that we hope will be of interest to both the industrial and academic communities.

5.1 Two Caveats

It is important to stress two points:

- We do not intend a literal analogy with Reduced Instruction Set Computers [16]. That paradigm was primarily motivated by the desire to increase processing speed while maintaining the cost, flexibility, and reliability of complex instruction sets. In our case we are primarily concerned with reducing costs and increasing reliability. We acknowledge that flexibility will generally be reduced along with intricacy in sensing and control.

- Our RISC paradigm is motivated by structured environments, where part geometry is known a priori and the emphasis is on reliable, repetitive operation. Certainly, more complex hardware will be required in unstructured environments such as hazardous waste sites or outer space, where little is known a priori. We expect that reliable software for such systems will continue to evolve and costs will decrease in the long run. We do not claim that simple hardware will solve all problems.

Acknowledgements

For discussions that helped us to refine this paradigm, we thank the participants in the NSF Workshop on Geometric Uncertainty in Motion Planning, in particular Brian Carlisle, Matt Mason, Jean-Claude Latombe, and Bud Mishra. Also, we thank our students, Anil Rao, Duk Kang, Aaron Wallack for helping to develop these examples, and Dan Whitney, Richard Wallace, Damian Lyons, and Howard Moraff for insightful feedback on our ideas.

References

[1] Rod Brooks. A layered intelligent control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2, 1986.

[2] Randy C. Brost. Automatic grasp planning in the presence of uncertainty. *The International Journal of Robotics Research*, December 1988. Also appeared in

Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, April, 1986.

[3] Brian Carlisle. Is u.s. robotics research of any use to u.s. industry? *IEEE Robotics and Automation Society Newsletter*, July 1991.

[4] S. H. Drake. *Using Compliance in Lieu of Sensory Feedback for Automatic Assembly*. PhD thesis, MIT, 1977.

[5] Michael A. Erdmann and Matthew T. Mason. An exploration of sensorless manipulation. In *IEEE International Conference on Robotics and Automation*, 1986. Also appears in IEEE Journal of Robotics and Automation, V. 4.4, August 1988.

[6] Ken Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, (to appear).

[7] Ken Goldberg, Matt Mason, and Ari Requicha. Nsf workshop on geometric uncertainty in robot motion planning: Summary report and bibliography. Technical Report 297, IRIS, August 1992. (The NSF-sponsored Workshop was held on Catalina Island, June 14-17, 1992).

[8] Kenneth Y. Goldberg. A kinematically-yielding gripper. In *22nd International Symposium on Industrial Automation*, October 1991. See U. S. Patent 5,098,145, granted March 26, 1992.

[9] Hajime Hitakawa. Advanced parts orientation system has wide application. *Assembly Automation*, 8(3), 1988.

[10] H. Inoue. Force feedback in precise assembly tasks. Memo 308, MIT Artificial Intelligence Laboratory, August 1974.

[11] Duk Kang and Ken Goldberg. Shape recognition by random grasping. In *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, July 1992.

[12] T. Lozano-Perez and J. L. Jones. Planning two-fingered grasps for pick-and-place operations no polyhedra. In *International Conference on Robotics and Automation*. IEEE, May 1990.

[13] Matthew T. Mason. Kicking the sensing habit. In *Asilomar Winter Workshop*. AAAI, November 1991.

[14] P. J. McKerrow. *Introduction to Robotics*. Addison-Wesley, 1991.

[15] James L. Nevins and Daniel E. Whitney. Computer-controlled assembly. *Scientific American*, 1978.

[16] D. Patterson and D. Ditzel. The case for the reduced instruction set computer. *Computer Architecture News*, October 1980.

[17] Michael A. Peshkin and Art C. Sanderson. Planning robotic manipulation strategies for workpieces that slide. *IEEE Journal of Robotics and Automation*, 4(5), October 1988.

[18] K. Pingle, R. Paul, and R. Bolles. *Programmable Assembly: Three Short Examples*. Stanford AI Lab Film, 1974.

[19] Anil Rao and Ken Goldberg. Grasping curved planar parts with a parallel jaw gripper. Technical Report 299, IRIS, August 1992. (submitted to the International Journal of Robotics Research).

[20] Anil Rao and Ken Goldberg. Shape from diameter: Strategies for recognizing polygonal parts. *IRIS Technical Report #292*, April 1992. (submitted to International Journal of Robotics Research).

[21] Andrew Tanzer and Ruth Simon. Why japan loves robots and we don't. *Forbes*, April 16 1990.

[22] Aaron S. Wallack and John F. Canny. Object localization using beam scanning. Unpublished memo. will use sparse resultant to find minimum least squared error fit. will use hash table for constant time determination, August 1992.

[23] Aaron S. Wallack and John F. Canny. Object localization using beam sensing. Technical report, UC Berkeley, 1992. three beam sensor: will use sparse resultant to find minimum least squared error fit. will suggest hash table for constant time determination.

[24] Daniel E. Whitney. 'real robots don't need jigs'. In *International Conference on Robotics and Automation*. IEEE, May 1986.

[25] I. M. Yaglom and V.G. Boltyanskii. *Convex Figures*. Holt, Rinehart and Winston, New York, 1961.