OpenArms: Towards Low-cost Robotic Manipulation

Morgan Quigley and **Reuben Brewer** and **Andrew Y. Ng**
Computer Science Department
Stanford University
353 Serra Mall
Stanford, California 94305

Abstract

In this paper, we discuss an approach by which we hope to make low-cost manipulators more easily available, thereby enabling robotic manipulation to scale to larger, more complex, and more pervasive deployments. We present a design approach which intends to produce a low-cost manipulator capable of performing useful tasks in unstructured environments. We provide historical motivation for this approach, and discuss the tradeoffs made in its design.

Introduction

Numerous challenges stand in the way of widespread deployment of general-purpose robots, ranging from theoretical topics in virtually every branch of artificial intellgence to more prosaic maintenance issues associated with complex robots. We propose that progress towards these challenges can be made, indirectly, by enlarging the pool of people with access to robots. We draw inspiration from numerous success stories in recent engineering practice, and believe that analogous steps could be made towards the development of "personal robots." More specifically, we propose that robotic hardware for is often overbuilt, and that a "good enough," or satisficing, approach to robotic hardware design may be beneficial in the drive for general-purpose robots.

Historical View

At a risk of oversimplifying engineering history, we briefly mention three separate instances to illustrate the case for lowering the cost of robotic manipulators.

First, the 1908 introduction of the Ford Model T revolutionized the automobile industry (Brooke 2008). It offered few features and the vehicle was chiefly designed to enable its mass production. However, it offered enough functionality, at a sufficiently low price point, that the nascent automotive industry was able to develop a loyal following. It should further be noted that a network of improved roads came *after* the invention of a low-cost automobile!

Closer to the topic at hand, the personal computers of the late 1970s deliberately reduced performance to allow their pricing to be widely accessible. The Apple II, particularly, employed numerous clever engineering feats in order to attain sufficient performance (Wozniak and Smith 2007). It was by no means the fastest computer of its day; minicomputers of the era were far more powerful, and far more expensive. Despite being inferior in every performance metric, the widespread deployment of the early personal computer allowed a huge audience the ability to experiment with computing. We note that VisiCalc, the first spreadsheet program for personal computers, was developed *after* their introduction. The use of personal computers in the workplace quickly became economically justifiable, but this was not obvious at the outset.

Finally, the affordability and availability of simple differential-drive robots such as the Pioneer series played a role in the recent advances in robotic mapping. Equipped with sonar or LIDAR sensors, these robots and similar minimalist platforms allowed robotic mapping and navigation algorithms to be readily reproducable in virtually any robotics laboratory. Furthermore, these platforms allowed for research into multi-robot mapping and control, which would have been prohibitively expensive with more expensive platforms.

We mention these historical developments to justify the design of simple and intentionally feature-limited manipulators. We also note that the "killer app" of a particular technology is often not known ahead of time, and premature optimization of a design for any particular performance criterion can make it cost-prohibitive for other, unforseen applications.

Driving Down Cost

One challenge (among many) currently limiting widespread deployment of robots is simple to express: robots are expensive. Many robotic manipulators used today for mobile manipulation experiments cost tens or hundreds of thousands of dollars. This cost presents a barrier to entry to the field, and promotes a "mainframe mentality" among researchers in a laboratory who must time-share access to the mobile manipulator. Risky experiments and field studies are often not performed, out of fear that robot downtime can lead to expensive repairs or impede the progress of other researchers sharing the robot.

Cost reduction would naturally follow production volume if and when robots find a place in the mass market. However, even before that occurs, we believe that significant cost reduction can occur without overly sacrificing perfor-

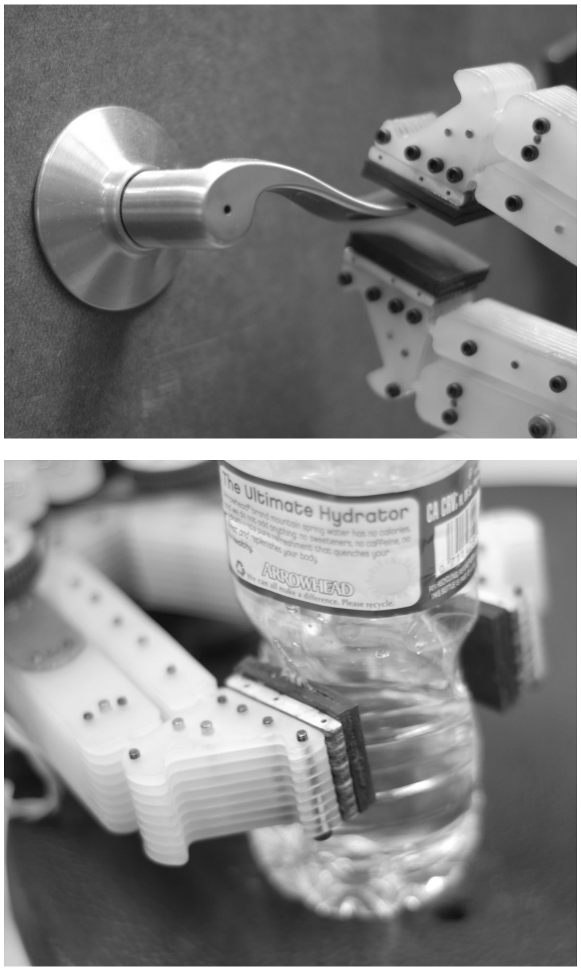

Figure 1: Unstructured manipulation tasks envisioned to be performed by service robots include door-opening and grasping common household objects. Pictured here only to illustrate that many such tasks can tolerate errors of a few millimeters, provided some measure of compliance exists in the mechanism or its control scheme.

mance on tasks performed by mobile manipulators in home and office environments. Moreover, we believe that it may even be *necessary* for cost reduction to occur before a mass-market application can be found for mobile manipulation. In some sense, complex general-purpose robots today are stuck in a chicken-and-egg situation: they are so expensive that economically justified mass-market applications are hard to imagine, yet without a mass-market application driving production volumes, current robots will likely remain expensive.

Our approach is intended to follow the historical examples mentioned in the previous section. Rather than push the limits of performance, the OpenArms project seeks to develop a manipulator at an accessible price point, yet which still offers enough functionality to perform useful tasks in a typical home or office. Such a manipulator will measure significantly worse on performance metrics frequently used to specify industrial manipulators, yet it may be buildable for a small fraction of the cost. We note that the manipulation tasks shown in Figures 1 require accuracies on the order of millimeters. Better accuracy and repeatability figures, while welcome, are not strictly required.

Scalable and Open Software

Scalable, open-source software takes the vision of low-cost hardware one step further: it can provide a powerful set of tools to a large community at essentially no cost. Our interest in large-scale open software stems from our work on the STanford AI Robot (STAIR) project, where we are seeking to develop the technology necessary to put a useful general-purpose robot in every home. As part of this research vision, we have spent significant time exploring the integration challenges common to large robotics projects. We have constructed several mobile platforms equipped with manipulators and various sensors, and performed demonstrations of various tasks which required significant interaction between various subfields of AI, such as fetching items from offices, using doors and elevators to navigate buildings, and unloading dishwashers (Quigley, Berger, and Ng 2007), (Klingbeil, Saxena, and Ng 2008), (Saxena, Driemeyer, and Ng 2007).

We have found that large numbers of people with various areas of expertise are required to implement solutions to these broad problems. Further, we found that software integration is a non-trivial issue, particularly when many layers of the software stack are undergoing simultaneous development. This problem is not unique to robotics, but it can be especially challenging when numerous computers and hardware devices must work in parallel to perform a particular task.

To address these challenges, we produced a series of software development frameworks, and have been working, together with colleagues at Willow Garage, on the Robot Operating System, ROS (Quigley et al. 2009). The central premise behind ROS, and similar frameworks such as YARP (Fitzpatrick, Metta, and Natale 2008), is that complex systems are more easily built as a collection of small programs, as opposed to a monolithic programming model. Interconnections between the small programs are managed automatically, providing transparent communications between programs running on various computers. Rigid modularity aids debugging and unit-testing of the complex systems which quickly arise in large-scale robotic integration.

Furthermore, open interfaces between the various software components allows for “hot-swapping” software modules. Coupled with logging and playback mechanisms, this allows for meaningful and reproducible performance comparisons between various algorithmic approaches at any layer of the software stack. Due to the open-ended nature of many AI problems, such flexibility is essential.

Availablity of high-quality, open-source robotics software further expands the pool of robotics experimenters by facilitating experimentation: any particular piece of the system can be examined, swapped out, and tested independently. Large-scale operational systems can be incrementally modified rather than built from scratch—for example, an alterna-

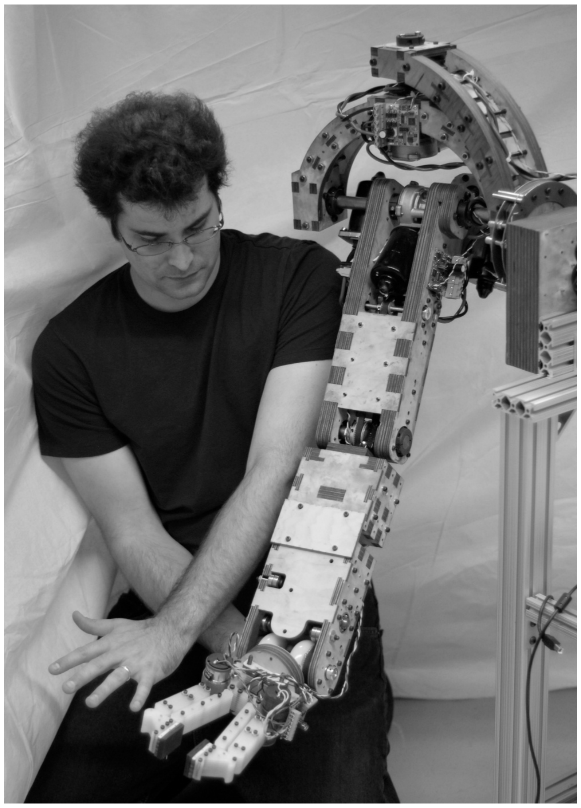

Figure 2: A prototype of the ideas discussed in this paper: a low-cost manipulator of roughly anthropomorphic size, built using readily-available or easily-machined parts.

tive grasp-point generation algorithm can be swapped into a working mobile manipulator.

OpenArms Prototype

To test these design concepts and try to define the "functionality floor," or the minimum set of capabilities required for useful manipulation in unstructured environments, we constructed the manipulator shown in Figure 2, an early version of which was shown at the 2009 IJCAI Robotics Workshop. The manipulator was designed with a budgetary goal of $1000 USD. It has roughly anthropomorphic dimensions and sufficient power, in some common configurations, to lift items such as drink bottles and common kitchen tools. Its design includes cheap, readily available, mass-produced mechanical components. Its structure consists of lasercut plywood and polypropylene plastic, a rapid-prototyping process available in many research laboratories as well as through web-based machine shops. In lieu of high-precision joint encoders, we employed 3D MEMS accelerometers to infer joint angles.

The overarching idea of the prototype is to shift complexity away from high-precision mechanisms and instead focus efforts on perception and control software. This is driven by the observation that complex software can be replicated with essentially no cost, whereas complex mechanisms often require high production volumes to be cheaply replicable.

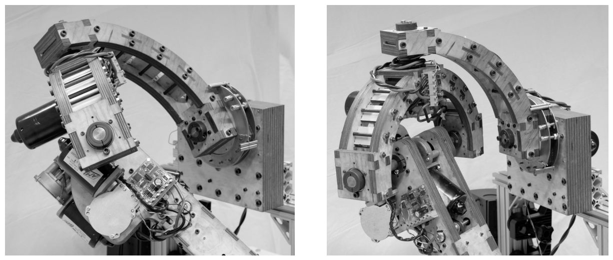

Figure 3: The shoulder is designed around the size constraints of commodity windshield-wiper motors, and uses a remote center of motion design to permit a wide range of motion while preserving the mechanical simplicity of a direct-drive configuration.

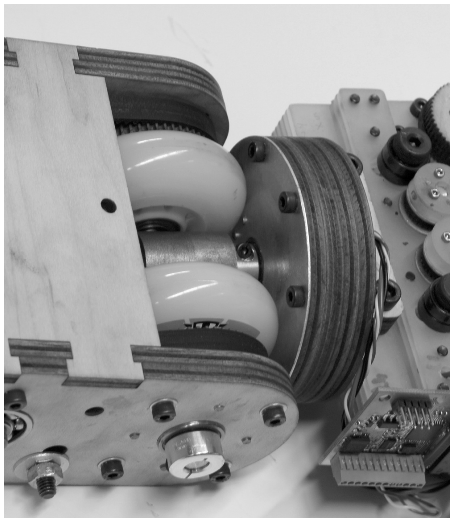

Figure 4: The wrist uses a friction-drive differential formed from rollerblade wheels and lasercut aluminum.

We employed mass-produced geared motors in direct-drive configurations where possible, to maximize mechanical simplicity. These motors, commonly used in automobile windshield wiper assemblies, are readily available and inexpensive. The shoulder was essentially designed around these motors, as shown in Figure 3.

The wrist is a friction-drive differential, using belt-driven roller-blade wheels as the load-bearing components. The wheels are tightly pressed against a lasercut aluminum plate, as shown in Figure 4. Because the compressed rubber wheels offer essentially zero backlash, this low-cost mechanism offers surprisingly high performance and torque transfer.

The gripper (Figure 5) is fabricated from lasercut polypropylene. Each finger is directly-driven by a low-cost gearmotor. To support parallel grasping tasks while incurring minimal cost, four-bar linkages are formed using flexures, or "living hinges."

Our choice of accelerometers for joint-angle sensing was by cost: due to their incorporation in mass-market products

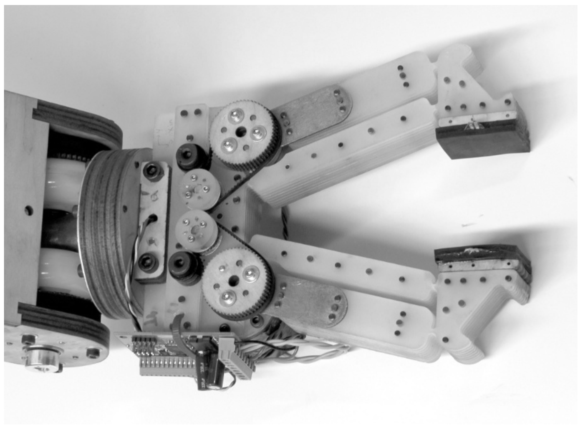

Figure 5: The gripper is fabricated from lasercut polypropylene, using flexures to create four-bar mechanisms. The small belts are used only to turn potentiometers to measure the finger positions; drive torques come from small gearmotors mounted directly below each finger.

such as cell phones and digital cameras, high-resolution accelerometers cost only a few dollars. We included an accelerometer in the design of our motor controllers (which fly on each link), and used their measurement of the direction of the gravity vector to infer the joint angle. An Extended Kalman Filter is used to combine knowledge of the kinematic parameters of the manipulator with the sensor stream, and thereby estimate the joint angles. Sensing singularities are present when a joint axis is vertical, and such situations must be avoided, and large accelerations or joint velocities can cause difficulties. Despite these limitations, we prototyped the low-cost accelerometer-based sensing scheme to more fully understand its capabilities and drawbacks.

All elements of our manipulator are open-source, including the mechanical and electrical designs as well as all firmware and software. We hope that this level of openness will spur further refinements and contributions, as all layers of the design are accessible, repairable, and tweakable.

The full technical design of our prototype can be found at http://openarms.stanford.edu

Challenges and Future Work

We intend to continue exploring low-cost manipulator design, seeking to define and implement the minumum set of features and performance metrics needed for various unstructured manipulation problems. An important direction for future work involves expanding the sensor suite of the prototype manipulator described in this paper. Although the accelerometer-only sensing scheme works well under static conditions, a straightforward implementation of this scheme, coupled with the “cogging” of commodity gearmotors, can lead to instabilities under high accelerations. In future work, we plan to enhance the high-dynamic performance of our prototype manipulators with joint encoders, as well as continue the process of improving the mechanical properties of the structures and joints.

References

Brooke, L. 2008. *Ford Model T: The Car That Put the World on Wheels*. Motorbooks.

Fitzpatrick, P.; Metta, G.; and Natale, L. 2008. Towards long-lived robot genes. *Robotics and Autonomous Systems* 56.

Klingbeil, E.; Saxena, A.; and Ng, A. 2008. Learning to open new doors. *Robotics Science and Systems (RSS) Workshop on Robot Manipulation*.

Quigley, M.; Berger, E.; and Ng, A. 2007. Stair: Hardware and software architecture. *AAAI Robotics Workshop*.

Quigley, M.; Gerkey, B.; Conley, K.; Faust, J.; Foote, T.; Leibs, J.; Berger, E.; Wheeler, R.; and Ng, A. Y. 2009. Ros: an open-source robot operating system. *Open-Source Software workshop of the IEEE International Conference on Robotics and Automation*.

Saxena, A.; Driemeyer, J.; and Ng, A. Y. 2007. A vision-based system for grasping novel objects in cluttered environments. *International Symposium on Robotics Research (ISRR)*.

Wozniak, S., and Smith, G. 2007. *iWoz: Computer Geek to Cult Icon: How I Invented the Personal Computer, Co-Founded Apple, and Had Fun Doing It*. W.W. Norton.