# ACME,
# A Telerobotic Active Measurement Facility

Dinesh K. Pai, Jochen Lang, John Lloyd and Robert J. Woodham
Department of Computer Science
University of British Columbia
Vancouver, Canada
{pai|jlang|lloyd|woodham}@cs.ubc.ca

**Abstract:** We are developing a robotic measurement facility which makes it very easy to build "reality-based" models, i.e., computational models of existing, physical objects based on actual measurements. These include not only models of shape, but also reflectance, contact forces, and sound. Such realistic models are crucial in many applications, including telerobotics, virtual reality, computer-assisted medicine, computer animation, computer games, and training simulators.

The Active Measurement Facility (ACME) is an integrated robotic facility designed to acquire a rich set of measurements from objects of moderate size[1], for building accurate physical models. ACME can provide precise motions of a test object; acquire range measurements with a laser range finder; position a 3-CCD color video camera, a trinocular stereo vision system, and other sensors around the object; probe the test object with a robot arm equipped with a force/torque sensor; and acquire registered measurements from all these sensors.

ACME is a telerobotic system with fifteen degrees of freedom. Everything in ACME, from force controlled probing to camera settings and lighting, is under computer control. We have also developed an extensive teleprogramming system for ACME. ACME is designed to be a shared resource, and can be controlled from any remote location on the Internet.

## 1. Introduction

We would like to make it very easy to build "reality-based" models, i.e., effective computational models of real, physical objects based on actual measurements. Applications in telerobotics, virtual reality, training simulators and computer-assisted medicine require such realistic models. Computer animations and games could also profit from more realistic models of objects. Such models should be sufficiently accurate for meaningful simulation and analysis but also efficient for interaction using graphics, haptics, and auditory displays. Recently, there has been significant progress in some areas such as modeling

[1]Typically less than 0.5m in diameter. Larger objects can be accommodated depending on the measurements to be acquired.

geometric shape from measurements (e.g., using Cyberware[2], and Hymarc's Hyscan system[3]).

However, the current state of automation for building models is inadequate in important ways. There are no systems that allow integrated modeling of physical attributes such as surface texture and friction, elastic deformation and contact force response, surface reflectance and other radiometric properties, and the sound field due to impact. Such models are essential for realistic simulation, and for rich visual, haptic and auditory interaction in virtual environments. Acquiring the data to build such models remains extremely tedious. This has resulted in widespread reliance on simple, idealized, mathematical models of objects which are never validated by comparison with real world objects.

There is thus a need for automated systems which can measure a large number of properties with reasonable accuracy, and combine these measurements into computational models. The Active Measurement Facility (ACME) is an integrated robotic facility designed to acquire these measurements from small objects. While sensor-rich robotic systems have been previously developed elsewhere (e.g., [1]), we believe this is the first integrated, automated measurement facility for building comprehensive models of everyday objects.

The remainder of the paper is organized as follows. In Section 2 we give a brief overview of ACME hardware facilities. The control architecture, including its novel Internet-based user interface, is detailed in Section 3. In Section 4 we describe teleprogramming for ACME. We conclude with a summary and outlook in Section 5.

## 2. System Overview

The ACME hardware can be conceptually divided into the following subsystems (see also Figure 1):

- A 3-DOF *Test Station*, on which a test object is placed and can be moved for presentation to the sensing equipment. It consists of two translational motion stages mounted at right angles and a rotation stage mounted on top. (All stages are made by Daedal, with linear and rotary accuracies of $\pm 0.00025$ in and $\pm 10$ arc-min.)

- A *Field Measurement System* (FMS) for measuring the light and sound fields around the test object (see Figure 2). Light field measurements can be used, for instance, to tabulate the BRDF (bi-directional reflectance distribution function) for rendering [5], or simply to get images. Sound fields can be used to build sound synthesis models [2]. A key component of the field measurement system is a high quality 3-CCD color video camera with computer controlled zoom and focus (Sony DXC 950). The camera and other sensors can be positioned with a 5-DOF gantry robot. A trinocular stereo vision system is also included in the FMS and is described below.

[2] `http://www.cyberware.com/`
[3] `http://www.hymarc.com/`

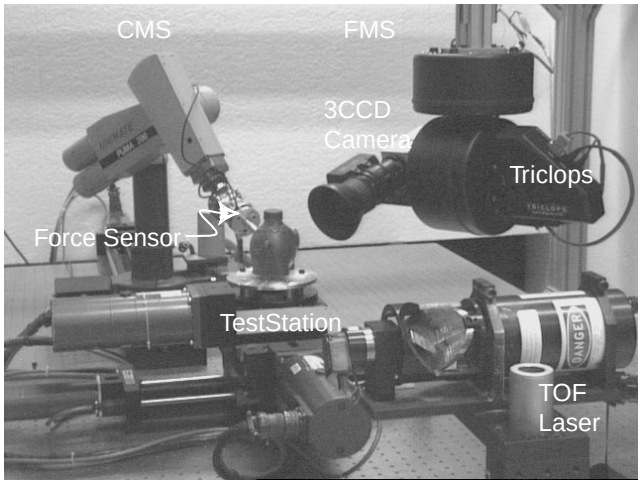
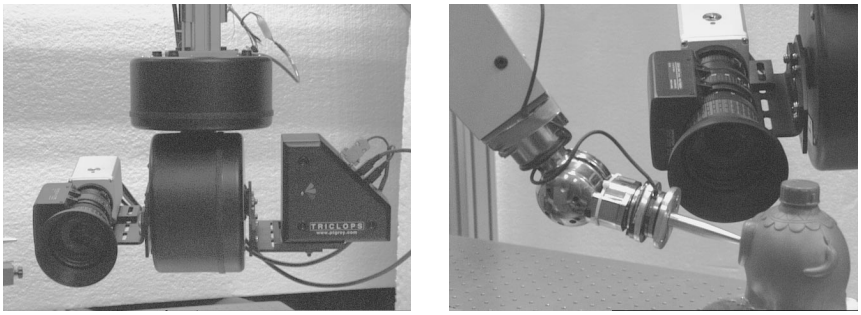



Figure 1. ACME Facility Overview



Field Measurement System (FMS) Force/Position Measurement (CMS)

Figure 2. ACME Subsystems

- A *Contact Manipulation System* (CMS), which includes a Puma 260 robot arm with a wrist mounted 6-axis force/torque sensor (ATI Mini 40). The CMS is intended to measure properties such as friction and stiffness, and to make controlled impacts to generate sounds. The arm is mounted on a long linear motion stage to increase its work space. Contact forces with the test object are controlled using active compliance. Figure 2 shows this subsystem acquiring contact force data from a test object's surface.

- Range measurement systems. Currently, we have two range measurement devices: a time-of-flight laser range finder (Acuity AccuRange 3000 LIR) and a Color *Triclops* stereo vision system from Point Grey Research [10]. Other approaches to shape measurements may be incorporated in the future. We would like to experiment with photometric stereo [14] utilizing

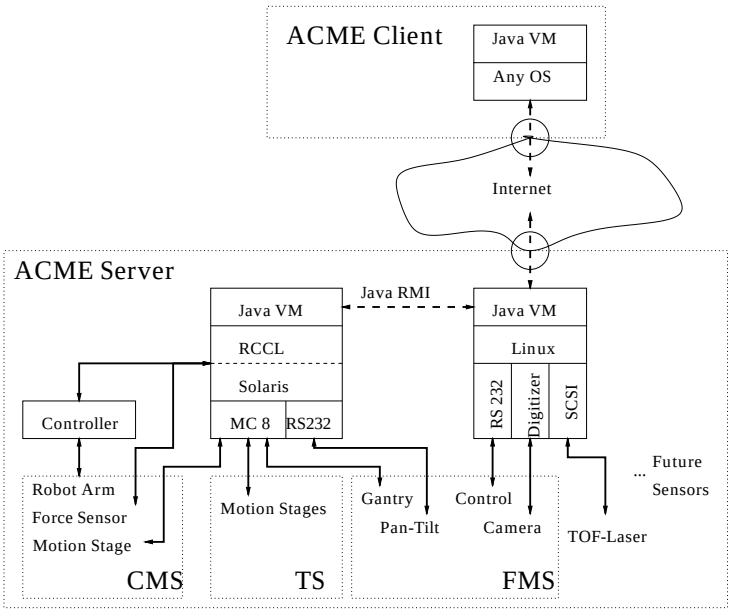



Figure 3. ACME Server: Overview of Control

the 3-CCD camera as the sensing device and appropriate light sources.

The design of ACME is directed at acquiring measurements for building models, and not the models themselves. However, ACME will provide nominal shape models, since these are necessary for obstacle avoidance, sensor planning, path planning, etc. The three-dimensional location and shape of objects on the test station are measured and registered relative to a facility-wide world coordinate frame.

- Miscellaneous subsystems, such as light sources, and control computers.

## 3. Control Architecture

In this section we describe the software and hardware architecture for controlling the ACME facility. The overall architecture is shown in Figure 3, and is divided into a server and a client. The robotic subsystems are controlled by the ACME Server (Section 3.1). Users interact with ACME using the client (Section 3.2). We discuss user level teleprogramming in Section 4.

### 3.1. ACME Server

The ACME server software consists of four layers. The highest layer is the user's `Experiment` which is described in the next section.

The next layer provides high level Java$^{TM4}$ objects called `Devices`, which are in turn subclassed into `Actuators` and `Sensors`. `Actuators` are robotic

[4]Java is a trademark of Sun Microsystems Inc., MountainView, Ca., USA

subsystems which can be controlled as a unit: these include the Test Station, the CMS (implemented using a Puma 260 robot arm), and the 5-DOF gantry robot ($x$-$y$-$z$ positioner plus 2-DOF pan-tilt head) comprising the FMS. `Sensor`s are sources of measurement data, such as the time-of-flight laser range sensor, 24-bit color camera, and the CMS's 6 axis force/torque sensor. All `Device` objects keep track of their state and spatial position. `Actuator` objects also have methods to solve their inverse kinematics.

An important feature of our design is that motions are first-class objects which can be directly manipulated or stored like any other object in the language. For instance, a motion can be handed to a simulator for verification or to a server for execution on the actual hardware. This is in contrast with typical robot programming languages in which motions are implicitly defined through procedures. The smallest motion object is a `Motion`, which is a generalized way to specify a specific motion of an `Actuator`. `Motion`s may also include a programmable impedance for those actuators capable of realizing this. Individual `Motion`s may be assembled into a `MotionPlan`, which takes care of sequencing motions of a given actuator and starting motions of different actuators at the same time.

The next layer of software generates real-time trajectories for each actuator device from a `MotionPlan`. This layer is implemented using the Robot Control C Library (RCCL) [7] with a Java$^{TM}$ front-end. RCCL generates (at 100 Hz) the joint setpoints necessary to realize the specified position and force commands (basing the latter on input from a 6-DOF force sensor). These joint setpoints in turn form the input to various controllers which comprise the lowest layer: the 6 joint servos in the PUMA controller, an 8 axis motion control card (Precision Micro Dynamics MC-8) controlling the Test Station and $x$-$y$-$z$ axes of the gantry, and a stepper motor interface for the gantry's pan-tilt head.

The grouping of physical devices into ACME subsystems has been described in 2, and it can also be seen from Figure 3. Physically, the system is distributed between several computers; separating low-level closed-loop control, trajectory generation, data acquisition, and networking.

### 3.2. ACME Client

The user interface of the ACME facility is through an ACME client. Figure 4 sketches the functionality of the client. The client serves as control terminal for the ACME facility; an important design feature is that the client can transparently connect to either a simulator of ACME, or to the real ACME facility.

The client consists of two major parts: the experiment window and the viewer. The experiment window is a simple development environment for designing ACME experiments. The viewer renders the state of the ACME facility (either real or simulated) and the state changes as an experiment proceeds. The complete ACME client is a 100% pure Java$^{TM}$ program enabling its use anywhere, on any system with an internet connection and Java$^{TM}$ virtual machine[5]

[5]However, the viewer utilizes the Java$^{TM}$3D API which is currently only available on a few systems.

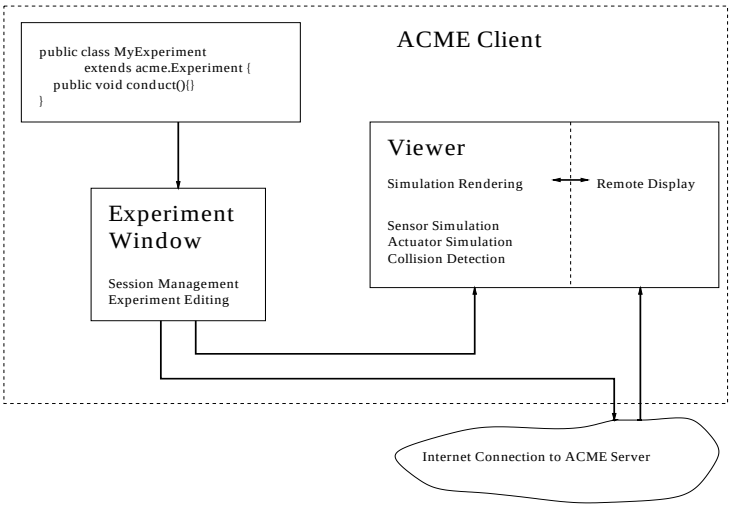



Figure 4. ACME Client: Functionality

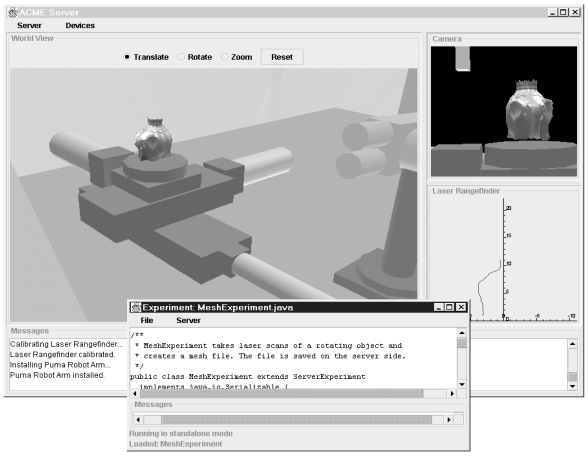



Figure 5. ACME Simulation on Client

The ACME client in stand-alone mode (connected to the simulator) is useful for design and verification of `Experiment`s, with the aid of a three-dimensional graphical simulation of the motion commands. In addition to motion, some of the sensors in ACME are simulated as well, using an object model. The model may be available from earlier experiments with ACME, or in fact may be any simple shape description file. Currently, the only sensors simulated are the TOF-Laser and camera. Since ACME allows not just one-shot measurements, but adaptive measurement and on-line model construction,

ACME `Experiments` can be quite complex; the simulation provides a way to test these algorithms safely. As an example, the simulator can return laser scan data and images from a geometric model of a test object, so that user algorithms for shape reconstruction can be tested in great detail.

The ACME client is also an Internet terminal to the ACME robotic device. Once a user is satisfied with the results of an experiment in simulation, the client may be connected to the ACME server and run the experiment on the real ACME facility as described in Section 4. While the ACME server executes the experiment, it sends back state information and data to the ACME client. The data may have nearly any format from "raw" sensor output to the result of any computation specified with the user's experiment and its helper objects. The client's viewer renders the state updates as received from the server, providing visual feedback through the graphical display of ACME. Note that an ACME client on a system without the Java$^{TM}$3d API can still upload an experiment to the ACME server but can not use the viewer to provide visual feedback of an experiment in progress.

## 4. Teleprogramming ACME

ACME is a high degree of freedom robotic system, and therefore requires a good programming environment to use it effectively. Several teleprogramming systems have been demonstrated in the literature [6, 12, 3] from which we have adapted programming ideas for ACME, as well as from the Cornell Mobot Scheme environment [11]. Experiments can be run on ACME from almost any site on the Internet. See [9, 4, 13] for other work on telerobotics on the Internet. We describe the programming environment of ACME below.

ACME programs are called "Experiments" and are written in the Java$^{TM}$ programming language. Our design exploits dynamic class uploading, security, multi-threading, and networking provided by Java$^{TM}$.

A simple example using the ACME1.0 API is shown below. A user-defined class (called `HelloRealWorld` here) moves the test object to a desired configuration, sets camera parameters, and takes a series of images of the test object from different angles around the object by rotating the test station. The image data is then stored on the client's local disk. An experiment is required to implement a `conduct` method which is called by the ACME server to conduct the user's experiment.

```
import acme.*;

public class HelloRealWorld
  extends ExperimentBase
  implements java.io.Serializable {

  private DeviceCom devCom = null;
  private Camera cam = null;
  private TestStation ts = null;

  //... other methods
```

```
  public void init() throws Exception {
    this.devCom = acme.getDeviceCom();
    this.ts = (TestStation) devCom.getDevice(TEST_STATION);
    this.cam = (Camera) devCom.getDevice(CAMERA);
    // set Camera state to desired settings
    CameraState camState = (CameraState) cam.getState();
    camState.zoom = 100; cam.setControl(camState);
    devCom.printMessage("Experiment initialized.");
    return;
  }

  public void conduct() {
    init();
    double[] jvals = new double[3];
    try {
      for (int deg = 0; deg < 360; deg += 45) {
        jvals[0] = 0; jvals[1] = 0;
        jvals[2] = deg / 180.0 * Math.PI;
        MotionPlan plan = new MotionPlan();
        Motion mot = plan.add(new Move (this.ts, jvals));
        plan.execute();
        while (mot.status() != Motion.COMPLETED) {
        Thread.sleep (1000);
        }
        ACMEImage image = this.cam.getImage();
        acme.saveData("image" + deg + ".ppm",
          ACMEImage.toBytePPM( image.getPixels(true), 640, 480));
      }
    }
    catch ( Exception e ) {
      devCom.printStatus( "Error in conduct()" );
      return;
    }
    return;
  }
}
```

---

The basic functionality for conducting experiments is defined by the interface `Experiment`, with some of the implementation provided by the abstract class `ExperimentBase`. In particular, it provides a reference to a special `acme` object. Various subsystems of ACME can be accessed through this reference; for instance the Test Station device is accessed invoking

```
acme.getDeviceCom().getDevice(TEST_STATION);
```

and an image can then be acquired by `cam.getImage()` and saved on the client's local disk using

```
acme.saveData("image" + deg + ".ppm",
 ACMEImage.toBytePPM(image.getPixels(true), 640, 480));
```

The client's Experiment class can be compiled on any computer. The ex-

periment is conducted by uploading the class into a Java$^{TM}$ Virtual Machine (JVM) running on the ACME server. The ACME server provides native implementations of methods which can access and control the ACME hardware (see Figure 3).

Some features of our design are worth mentioning, since they make it easy to develop correct experiments and run them easily from a remote location.

First, while it would be possible, and conceptually simple, to use Remote Method Invocation (RMI) to directly teleoperate ACME from a remote site, the high latency and low bandwidth of the Internet makes this approach problematic, particularly for experiments involving contact with the test object. To address this problem, in previous work we have developed a system for model-based telerobotic control, and demonstrated its feasibility with a similar robot system [6, 8]. In ACME, we also use a teleprogramming approach, and send small robot programs (ACME `Experiments`) instead of low-level motion and sensing commands. This allows feedback loops to be closed at the remote ACME server and enables adaptive measurement techniques such as view-point planning.

Second, the ability to load classes at run time makes it possible to simulate the experiment by loading *exactly the same compiled bytecode* into a JVM on the user's computer. In this case, a different implementation of ACME classes and their methods are loaded into the JVM, and provide a graphical simulation of ACME (see Figure 5). Therefore, errors in the experiment can be caught early, and the simulated experiment can be immediately run on the real ACME facility, with no recompilation.

Third, dynamic class loading is well suited for this type of teleprogramming. The `Experiment` object is instantiated on the client, serialized, and sent to the ACME server, where it is re-instantiated. This has two benefits: (a) It allows interactive experimentation. The user can modify an Experiment class (for instance based on measurements returned from ACME), and reload it into a running ACME server. (b) It allows us to develop a library of idiomatic experiment objects which can be customized on the client (for instance based on user manipulation of the graphical simulation of ACME).

## 5. Conclusions

We are developing a telerobotic measurement facility, ACME, with goal of making it extremely easy to build more accurate and complete physical models of everyday objects. When completed, ACME will be able to acquire a large number of carefully registered measurements of a given object including shape, reflectance, sound, and contact forces. ACME is also a high degree of freedom robotic system with a complex array of sensors; we have developed an extensive teleprogramming architecture for programming this system from any location on the Internet.

**Acknowledgments**

Several people have contributed significantly to the development of ACME at the University of British Columbia; without their efforts the system could not

have been built. They are, in alphabetical order: R. Barman, C. Chiu, J. Fong, A. Fournier, L. Ke, S. Kingdon, and A. K. Mackworth. Financial support was provided in part by grants from NSERC and IRIS NCE. Lang and Lloyd were supported in part by NSERC fellowships. We would also like to thank Point Grey Research for their support and assistance with the Triclops system.

## References

[1] G. Hirzinger, B. Brunner, J. Dietrich, and J. Heindl, "Sensor-Based Space Robotics – ROTEX and Its Telerobotic Features". *IEEE Transactions on Robotics and Automation*, October 1993, pp. 649–663 (Vol. RA-9, No. 5).

[2] K. van den Doel and D. K. Pai, "The Sounds of Physical Shapes," *Presence*, The MIT Press, 1998, pp. 382-395, (Vol. 7, No. 4).

[3] J. Funda, T. S. Lindsay, and R. P. Paul, "Teleprogramming: Toward delay-invariant remote manipulation". *Presence*, Winter 1992, pp. 29–44 (Vol. 1, No. 1).

[4] K. Goldberg, M. Mascha, S. Gentner, N. Rothenberg, C. Sutter and J. Wiegley, "Desktop Teleoperation via the World Wide Web". *Proceedings 1995 IEEE International Conference on Robotics and Automation*, Nagoya, Japan, May, 1995, pp. 654–659.

[5] Paul Lalonde and Alain Fournier, "Generating Reflected Directions from BRDF Data". *Computer Graphics Forum, Special issue on Eurographics '97*, August 1997, pp. 293–300, (Vol. 16, No. 3).

[6] J. E. Lloyd, J. S. Beis, D. K. Pai, and D. G. Lowe, "Model-based Telerobotics with Vision". *Proceedings 1997 IEEE International Conference on Robotics and Automation* Albuquerque, NM, April 1997, pp. 1297–1304.

[7] John E. Lloyd and Vincent Hayward, "Multi-RCCL User's Guide". McGill CIM, April, 1992.

[8] J. E. Lloyd, and D. K. Pai, "Extracting Robotic Part-mating Programs from Operator Interaction with a Simulated Environment." In the proceedings of the *Fifth International Symposium on Experimental Robotics*, (Barcelona), June 1997.

[9] E. Paulos and J. Canny, "Delivering Real Reality to the World Wide Web via Telerobotics". *Proceedings 1996 IEEE International Conference on Robotics and Automation*, Minneapolis, Minnesota, April 1996, pp. 1694–1699.

[10] Triclops On-line Manual, *http://www.ptgrey.com/*, Point Grey Research, Vancouver, Canada.

[11] J. Rees and B. Donald. Program mobile robots in scheme. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, Nice, France, 1992, pp. 2681–2688.

[12] C. R. Sayers, "Operator Control of Telerobotic Systems for Real World Intervention". Ph. D. thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104 USA, 1995.

[13] K. Taylor and J. Trevelyan, "Australia's Telerobot On The Web". *26th International Symposium On Industrial Robots*, Singapore, October 1995. `http://telerobot.mech.uwa.edu.au/`.

[14] R. J. Woodham, "Gradient and Curvature from the Photometric-Stereo Method, Including Local Confidence Estimation". *Journal of the Optical Society of America A*, November 1994, pp. 3050-3068, (Vol. 11, No. 11).