Published at:
2004 IEEE Conference on Robotics, Automation, and Mechatronics (IEEE-RAM), Dec. 2004, Singapore, pp. 446-451 (6)

The Autonomous Underwater Vehicle Initiative – Project Mako

Thomas Bräunl, Adrian Boeing, Louis Gonzales, Andreas Koestler, Minh Nguyen, Joshua Petitt
The University of Western Australia
EECE – CIIPS – Mobile Robot Lab
Crawley, Perth, Western Australia
tb@ee.uwa.edu.au

***Abstract*—We present the design of an autonomous underwater vehicle and a simulation system for AUVs as part of the initiative for establishing a new international competition for autonomous underwater vehicles, both physical and simulated. This paper describes the competition outline with sample tasks, as well as the construction of the AUV and the simulation platform.** [1]

Keywords—autonomous underwater vehicles; simulation; competition; robotics

I. INTRODUCTION

Mobile robot research has been greatly influenced by robot competitions for over 25 years [2]. Following the MicroMouse Contest and numerous other robot competitions, the last decade has been dominated by robot soccer through the FIRA [6] and RoboCup [5] organizations. Most of these competitions involved wheeled robots on well defined flat surfaces, with legged robots only introduced a few years ago.

Choosing a competition environment similar to conditions found in industrial applications will improve the relevance of the research work performed. We intend to create a new robotics initiative for autonomous underwater vehicles (AUVs). This is a both interesting and challenging research area with direct commercial applications and existing industries in the Australian and Asian-Pacific region. The initiative shall comprise competitions tied in with international conferences, as well as design guidelines for real AUVs and a public domain AUV simulation system (SubSim), developed by the authors.

A related underwater competition has been organized for a number of years by the Association for Unmanned Vehicle Systems International (AUVSI) in North America [1]. Our proposed competition concentrates on the Australasian region, will introduce a number of different competition events, and will create a common simulation platform available for teams that cannot afford to build an AUV.

[1] *Raytheon Australia has initiated and sponsored the upcoming Australian AUV competition. Part of the research presented in this paper has been sponsored by Raytheon Australia.*

II. AUV COMPETITION

The AUV Competition will comprise two different streams with similar tasks

- Physical AUV competition and
- Simulated AUV competition

Competitors have to solve the same tasks in both streams. While teams in the physical AUV competition have to build their own AUV (see Chapter 3 for a description of a possible entry), the simulated AUV competition only requires the design of application software that runs with the SubSim AUV simulation system (see Chapter 4).

The competition tasks anticipated for the first AUV competition (physical and simulation) to be held around July 2005 in Perth, Western Australia, are described below. All events will take place in a (physical or simulated) swimming pool of Olympic size, whenever possible. Specific rules regarding points being awarded, timing, etc., will be published at a later stage.

Task 1: Wall Following

The AUV is placed close to a corner of the pool.

The task is to follow the pool wall without touching it. The AUV should perform one lap around the pool, return to the starting position, then stop.

Task 2: Pipeline Following

A plastic pipe (diameter and color to be specified) is placed along the bottom of the pool, starting on one side of the pool and terminating on the opposite side. The pipe is made out of straight pieces and 90 degree angle pieces. The AUV is placed over the beginning of the pipe close to a side wall of the pool.

The task is to follow the pipe on the ground until the opposite wall has been reached.

Task 3: Target Finding

The AUV is placed close to a corner of the pool. A target plate with a distinctive texture (size and color to be specified) will be placed at a random position within a 3m diameter from the center of the pool.

The task is to find the target plate in the pool, drive the AUV directly over it and drop an object on it from the AUV's payload container. (AUV's without the ability to drop a payload should just hover over the target plate).

Task 4: Object Mapping

A number of simple objects (balls and boxes, color and sizes to be specified) will be placed near the bottom of the pool, distributed over the whole pool area. The AUV starts in a corner of the pool.

The task is to survey the whole pool area, e.g. using a sweeping pattern, and record all objects found at the bottom of the pool. The AUV shall return to its start corner and stop there. In a subsequent data upload (AUV may be taken out of the pool and a cable attached to its onboard system), a map of some format should be uploaded that shows location and type (ball or box) of all objects found by the AUV.

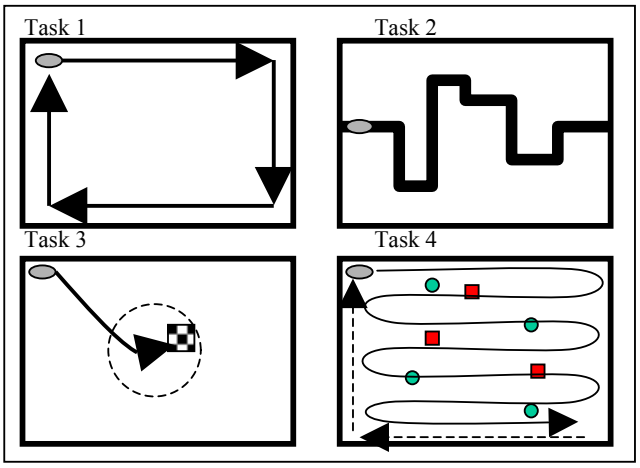

Figure 1. Competition tasks

III. AUV HARDWARE

For constructing a new AUV prototype from scratch, the mechanical and electrical systems of the AUV, as well as its sensor equipment had to be designed and considered simultaneously. This chapter presents a summary of the design of the *Mako* AUV.

A. Mechanical and Electrical Design

A two-hull, four-thruster configuration was chosen for the final design because of its significant advantages over competing design proposals (see Figure 2). This design provided the following:

- Ease in machining and construction due to its simple structure
- Relative ease in ensuring watertight integrity because of the lack of rotating mechanical devices such as bow planes and rudders
- Substantial internal space owing to the existence of two hulls
- High modularity due to the relative ease with which components can be attached to the skeletal frame
- Cost-effectiveness because of the availability and use of common materials and components
- Relative ease foreseen in software control implementation in using a four thruster configuration when compared to using a ballast tank and single thruster system
- Acceptable range of motion provided by the four thruster configuration
- Ease in submerging with two vertical thrusters
- Static stability due to the separation of the centers of mass and buoyancy, and dynamic stability due to the simple alignment of thrusters with easily identifiable centers of drag

Precision in controlling the vehicle and overall simplicity were pursued in the mechanical design over speed and sleekness. Although speed and sleekness are desirable qualities in AUVs, there is no real need for these as per the tasks required for the competition.

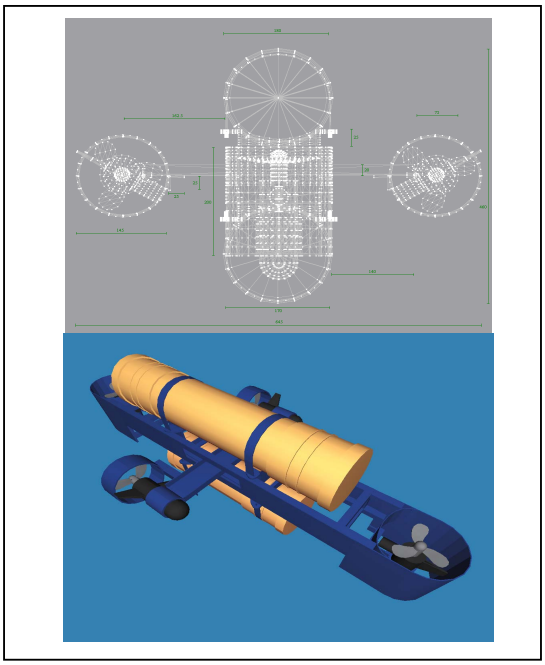

Figure 2. Design of AUV Mako

Simplicity and modularity were key goals in both the mechanical and electrical system designs. The mechanical design was made symmetrical to not only make construction relatively straightforward, but to also simplify software modeling. The materials chosen were common, cost-effective and easily machineable. Only crucial electronic components were purchased with improvisations made for other components. With the vehicle not intended for use under more than 5 m of water, pressure did not pose a major problem. The *Mako* (Figure 4) measures 1.34 m long, 64.5 cm wide and 46 cm tall.

The vehicle comprises two watertight hulls machined from PVC separately mounted to a supporting aluminum skeletal frame. Two thrusters are mounted on the port and starboard sides of the vehicle for longitudinal movement, while two others are mounted vertically on the bow and stern for depth control. The *Mako's* vertical thruster diving system is not power conservative, however, when a comparison is made with ballast systems that involve complex mechanical devices, the advantages such as precision and simplicity that comes with using these two thrusters far outweighs those of a ballast system.

Propulsion is provided by four modified 12V, 7A trolling motors that allow horizontal and vertical movement of the vehicle. These motors were chosen for their small size and the fact that they are intended for underwater use; a feature that minimized construction complexity substantially and provided watertight integrity.

The starboard and port motors provide both forward and reverse movement while the stern and bow motors provide depth control in both downward and upward directions. Roll and pitch is passively controlled by the vehicle's innate righting moment, though pitch can be controlled by the stern and bow motors if necessary. Lateral movement is not possible for the vehicle, however, this ability is not necessary as the starboard and port motors allow yaw control which permits movement in any global horizontal direction. Overall, this provides the vehicle with 4DOF that can be actively controlled. These 4DOF provide an ample range of motion suited to accomplishing a wide range of tasks.

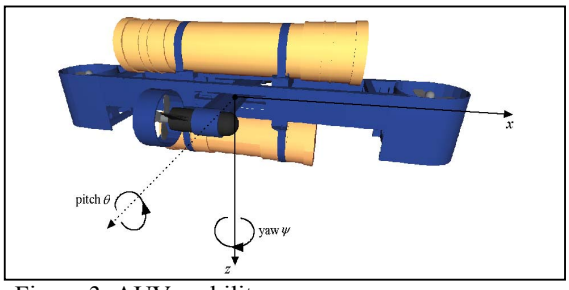

Figure 3. AUV mobility

B. Controllers

The control system of the *Mako* is separated into two controllers; an Eyebot microcontroller and a mini-itx PC running Linux. The Eyebot controller runs at 33MHz and comprises 512K of ROM as well as 2048K of RAM. This controller's primary purpose is in controlling the four thrusters, that is, controlling the vehicle's movement and the AUV's sensors. The mini PC comprises a Cyrix 233MHz processor, 32Mb of RAM and a 5GB hard drive. Its function is to provide processing power for the computationally intensive vision system.

Since the thrusters would be in continuous use while vision and sonar systems were operational, it was logical to distribute functions and control over two controllers. This allows a more powerful and concurrent approach to dealing with data and controlling the vehicle.

Motor controllers designed and built specifically for the thrusters provide both speed and direction control. Each motor controller interfaces with the Eyebot controller via two servo ports. Due to the high current used by the thrusters, each motor controller produces a large amount of heat. To keep the temperature inside the hull from rising too high and damaging electronic components, a heat sink attached to the motor controller circuit on the outer hull was devised. Hence, the water continuously cools the heat sink and allows the temperature inside the hull to remain at an acceptable level.

C. Sensors

The sonar/navigation system utilizes an array of Navman Depth2100 echo sounders. One of these is used to provide a crude, but effective depth sensor. An echo sounder facing downwards on the vehicle allows for depth feedback if the maximum water depth is known.

A low-cost Vector 2X digital magnetic compass module provides for yaw or heading control. The small module delivers high accuracy and interfaces with the Eyebot controller via digital input ports.

A simple dual water detector circuit connected to analogue-to-digital converter (ADC) channels on the Eyebot controller is used to detect the unlikely but possible event of a hull breach. Two probes run along the bottom of each hull. This allows for the location (upper or lower hull) of the leak to be known, therefore saving time in isolating a leak. The Eyebot periodically monitors whether or not the hull integrity of the vehicle has been compromised, and if so immediately surfaces the vehicle.

To ensure adequate power is being supplied to the vehicle, particularly to the motors, a simple power monitor is used to detect when the voltage level drops to unacceptable levels. This is essential as a low voltage will result in the thrust characteristics of the motors drifting from expected values, therefore compromising accurate control of the vehicle. The power monitor is essentially a voltage divider circuit that connects to an ADC channel on the Eyebot. This allows the Eyebot to periodically monitor the supply voltage and to surface the vehicle when it senses a low voltage.

The sonar system for the *Mako* AUV consists of four 200 kHz depth sounder transducers that are used for boat depth sensors. These transducers are ideal for use with the AUV as they are relatively cheap and are capable of performing depth sensing within a range that is similar to many more expensive digital altimeters.

The transducers are placed on the bow, port and starboard sides of the AUV and these will be used to determine the proximity of obstacles to the AUV. One sensor is pointing directly down for depth sensing, as mentioned before. Currently, there are only provisions for four of these sounder transducers but more transducers can be employed as required with only minimal change to the current design. The future use of more sensors can be used

to create an omni-directional view of the surrounding environment.

Each of these transducers are linked to a control circuit board that provides transmitter/receiver capabilities as well as timing capabilities to enable each sensor to determine the distance the AUV is from an obstacle in front of the sensor. All data lines from the sensors are connected to timing processor inputs (TPU channels) of the EyeBot controller. As all the sensors emit and receiver a 200 kHz frequency signal, there is a need to prevent collisions of signals and erroneous reception of another sensor's signal. This contention issue is resolved by serializing sonar measurements through the Eyebot controller.

Figure 4. AUV Mako

IV. AUV SIMULATION

There are many advantages to having a simulation system for the AUV competition. Simulation allows concurrent development of the hardware platform and control software, reducing the time required to build a complete solution. It also enables groups who have financial or physical limitations (such as the lack of an appropriately sized body of water) to compete. Finally, simulations allow far greater control over the environment in which the submarine is placed. This simplifies the testing of a program's performance in varied conditions.

A. Software design

The simulation software is designed to address a broad variety of users with different needs, such as the structure of the user interface, levels of abstraction, and the complexity of physical and sensor models. As a result, the most important design goal for the software is to produce a simulation tool that is as extensible and flexible as possible.

Therefore, the entire system was designed with a plug-in based architecture from the ground up. Entire components, such as the end-user API, the user interface and the physical simulation library can be exchanged to accommodate the users' needs. This allows the user to easily extend the simulation system by adding custom plug-ins written in any language supporting dynamic libraries, such as standard C or C++.

Further goals include ease of use, portability, execution speed, and the resulting size of the distribution binary. Figure 5 shows the basic architecture of the simulation system. The simulation system provides a software developer kit (SDK) that contains the framework for plug-in development, and tools for designing and visualizing the submarine. The software packages used to create the simulator include:

wxWidgets [8] (formerly wxWindows) – A mature and comprehensive open source cross platform C++ GUI framework. This package was selected as it greatly simplifies the task of cross platform interface development. It also offers straightforward plug-in management and threading libraries.

TinyXML [9] is a C++ XML parser. It was chosen over other XML parsing packages as it is simple to use, and is small enough to distribute with the simulation.

The user can select the physics libraries used by the simulator. However the default physics simulation system employed is the Newton Game Dynamics engine [10], which implements a fast and stable deterministic solver.

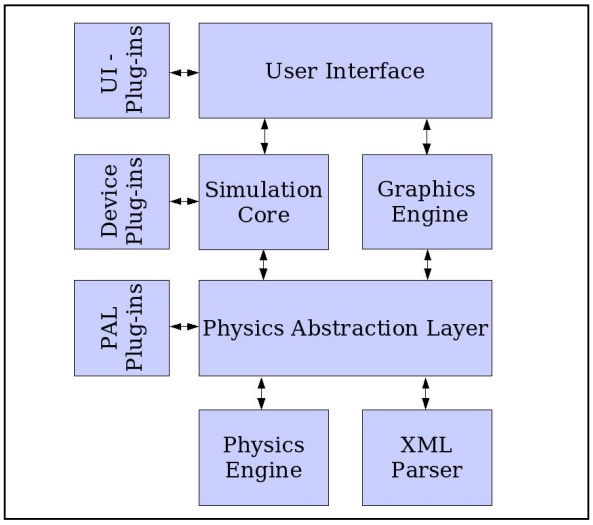

Figure 5. SubSim software design

B. Physics Models

The underlying low-level physical simulation library is responsible for calculating the position, orientation, forces, torques and velocities of all bodies and joints in the simulation. Since the low-level physical simulation library performs most of the physical calculations, the higher-level physics abstraction layer (PAL) is only required to simulate motors and sensors.

The PAL allows custom plug-ins to be incorporated to the existing library, allowing custom sensor and motor models to replace, or supplement the existing implementations.

1) Propulsion Model

The motor model implemented in the simulation is based off the standard armature controlled DC motor model [11]. The transfer function for the motor in terms of an input voltage (V) and output rotational speed (θ) is:

$$\frac{\theta}{V} = \frac{K}{(Js+b)(Ls+R)+K^2}$$

Where:

J is the moment of inertia of the rotor,

b is the damping ratio of the mechanical system,

L is the rotor electrical inductance,

R is the terminal electrical resistance,

and K is the electro motive force const

The default thruster model implemented is based on the lumped parameter dynamic thruster model developed by D. R. Yoerger et al. [7].

The thrust produced is governed by:

Thrust = $C_t \Omega |\Omega|$

Where

Ω is the propeller angular velocity,

and C_t is the proportionality constant.

Simulation of control surfaces is also supported. The model used to determine the lift from diametrically opposite fins [12] is given by:

$$L_{fin} = \frac{1}{2} \rho C_{L_{\delta f}} S_{fin} \delta_e v_e^2$$

Where: L_{fin} is the lift force,

ρ is the density,

$C_{L_{\delta f}}$ is the rate of change of lift coefficient with respect to fin effective angle of attack,

S_{fin} is the fin platform area,

δ_e is the effective fin angle,

v_e is the effective fin velocity

2) Simple Propulsion Model

One of the design goals of the simulation system is to ensure ease of use. To assist in accomplishing this goal, a much simpler model for the propulsion system is also provided, in the form of an interpolated look-up table. This allows a user to experimentally collect input values and measure the resulting thrust force, applying these forces directly to the submarine model.

3) Sensor Models

The PAL already simulates a number of sensors. Each sensor can be coupled with an error model to allow the user to simulate a sensor that returns data similar to the accuracy of the physical equipment they are trying to simulate. Many of the positional and orientation sensors can be directly modeled from the data available from the lower level physics library. Every sensor is attached to a body that represents a physical component of an AUV.

The simulated inclinometer sensor calculates its orientation from the orientation of the body that it is attached to, relative to the inclinometers own initial orientation. Similarly, the simulated gyroscope calculates its orientation from the attached body's angular velocity, and its own axis of rotation. The velocimeter calculates the velocity in a given direction from its orientation axis and the velocity information from the attached body.

Contact sensors are simulated by querying the collision detection routines of the low-level physics library for the positions where collisions occurred. If the collisions queried occur within the domain of the contact sensors, then these collisions are recorded.

Distance measuring sensors, such as echo-sounders and Positional Sensitive Devices (PSDs) are simulated by traditional ray casting techniques, provided the low level physics library supports the necessary data structures.

A realistic synthetic camera image is being generated by the simulation system as well. With this, user application programs can use image processing for navigating the simulated AUV. Camera user interface and implementation are similar to the EyeSim mobile robot simulation system [3].

4) Environment Model

Detailed modeling of the environment is necessary to recreate the complex tasks facing the simulated AUV. Dynamic conditions force the AUV to continually adjust its behavior. E.g. introducing (ocean) currents causes the submarine to permanently adapt its position, poor lighting and visibility decreases image quality and eventually adds noise to PSD and vision sensors.

The terrain is an essential part of the environment as it defines the universe the simulation takes part in as well as physical obstacles the AUV may encounter.

5) Error Models

Like all the other components of the simulation system, error models are provided as plug-in extensions. All models either apply characteristic, random, or statistically chosen noise to sensor readings or the actuators control signal.

We can distinguish two different types of errors: Global errors and local errors. Global errors, such as voltage gain, affect all connected devices. Local errors only affect a certain device at a certain time. In general local errors can be data dropouts, malfunctions or device specific errors that occur when the device constraints are violated. For example, the camera can be affected by a number of errors such as detector, Gaussian, and salt and pepper noise. Voltage gains (either constant or time dependent) can interfere with motor controls as well as sensor readings.

Peculiarities of the medium the simulation is running in have to be considered, e.g. refractions due to glass/water transitions and condensation due to temperature differences on optical instruments inside the hull.

C. API

The simulation system implements two separate application programmer interfaces (APIs). The first API is the internal API, which is exposed to developers so that they can encapsulate the functionality of their own controller API. The second API is the RoBIOS API, a user friendly API that mirrors the functionality present on the Eyebot [4] controller found on the Mako.

The internal API consits of only five functions:

```
SSID InitDevice(char *device_name);
SSERROR QueryDevice (SSID device, void *data);
SSERROR SetData(SSID device, void *data);
SSERROR GetData(SSID device, void *data);
SSERROR GetTime(SSTIME time);
```

The function InitDevice initializes the device given by its name and stores it in the internal registry. It returns a unique handle that can be used to further reference the device (e.g. sensors, motors).

QueryDevice stores the state of the device in the provided data structure and returns an error if the execution failed.

GetTime returns a time stamp holding the execution time of the submarines program in ms. In case of failure an error code is returned.

The functions that are actually manipulating the sensors and actuators and therefore affect the interaction of the submarine with its environment are either the GetData or SetData function.

While the first one retrieves the data (e.g. sensor readings) the later one changes the internal state of a device by passing control and/or information data to the device. Both return appropriate error codes if the operation fails.

D. Example

The following example is written using the RoBIOS API [4] and lets an AUV perform wall-following along its left side. The submarine parameters, its sensors, the whole simulation environment, as well as the simulation settings are specified by XML files.

```
#include <eyebot.h>

int main() {
  PSDHandle psd;
  MotorHandle left_motor;
  MotorHandle right_motor;
  int distance, motor_speed;

  /* init the motors */
  left_motor= MOTORInit( LEFT_MOTOR );
  right_motor=InitDevice(RIGHT_MOTOR);
  /* init and start PSD, set cont. measurement */
  psd=PSDInit (LEFT_PSD);
  PSDStart( psd, TRUE );
  MOTORDrive(right_motor, 50); /* medium speed */
  MOTORDrive(left_motor,  50);

  while(1)
  { distance = PSDGet(psd);  /* dist. to left */
    if (distance < 100)      /* too close */
      MOTORDrive(left_motor,  90); /* turn R */
    else if (distance > 200)      /* too far */
      MOTORDrive(right_motor, 90); /* turn L */
    else          /* straight with med. speed */
    { MOTORDrive(right_motor, 50);
      MOTORDrive(left_motor,  50);
    }
  }
}
```

As mentioned above, the actual program is coded using RoBIOS as API, however any API could be used. This application can be compiled using a common C compiler. RoBIOS is designed to work in the simulation as well as on the actual robot. Note that for simplicity error checking is omitted.

V. Conclusions

We have presented the outlines of a new robotics challenge, an AUV competition for both physical and simulated AUVs. These systems require numerous onboard sensors and real-time navigation and decision systems. The AUV competition will be a challenge for years to come.

Our prototype AUV gives a guideline for interested parties to build their own AUV, while the description of the simulation system gives a preview of the system that will be used to conduct the AUV simulation event. SubSim offers a powerful, comprehensive, yet easy to use simulation system for a variety of applications and users. The AUV competition is just one application of the simulation system. Due to our design philosophy, SubSim offers high extensibility, flexibility and usability for AUV simulation.

References

[1] AUVSI and ONR's 7th International Autonomous Underwater Vehicle Competition, http://www.auvsi.org/ competitions/water.cfm

[2] T. Bräunl, *Research relevance of Mobile Robot Competitions*, IEEE Robotics and Automation Magazine, vol. 6, no. 4, Dec. 1999, pp. 32-37 (6)

[3] T. Bräunl, Embedded Robotics – Mobile Robot Design and Applications with Embedded Systems, Springer-Verlag, Heidelberg, 2003

[4] T. Bräunl, UWA Mobile Robot Lab, http://robotics.ee.uwa.edu.au

[5] Proceedings of 7th RoboCup International Symposium, Padua, Italy, 2003

[6] Proceedings of the FIRA Robot World Congress 2003, Wien, Oct. 2003

[7] D. R. Yoerger, J. G. Cooke, J. E. Slotine, "The Influence of Thruster Dynamics on Underwater Vehicle Behavious and Their Incorporation Into Control System Design," IEEE J. Oceanic Eng., vol 15, no. 3, pp. 167-178, 1991.

[8] WX Widgets, http://www.wxwidgets.org/

[9] Tiny XML, http://tinyxml.sourceforge.net/

[10] Newton Game Dynamics Engine, http://www.physicsengine.com/

[11] R.C. Dorf, R.H. Bishop, *Modern Control Systems*, Prentice-Hall, 2001, Ch. 4, pp174-223.

[12] P. Ridley, J. Fontan, P. Corke, *Submarine Dynamic Modeling*, Australasian Conference on Robotics and Automation, 2003, CD-ROM Proceedings, ISBN 0-9587583-5-2