2001 IEEE/ASME International Conference on
Advanced Intelligent Mechatronics Proceedings
8—12 July 2001 • Como, Italy

A Complex Mechatronic System: from Design to Application

Nicola Tomatis[†], Roberto Brega[‡], Kai Arras[†], Björn Jensen[†], Benoit Moreau[†], Jan Persson[†], Roland Siegwart[†]

Abstract – **Progress in mobile robotics requires the researchers to access and improve all modules that compose the robot, from low-level mechanical components to high-level reasoning systems. This paper presents the development process of the robots built at the Autonomous Systems Lab, EPFL Lausanne, Switzerland. Starting from the mechanical and electrical design up to the application, we show the challenges that needed to be faced as well as the solutions that have been devised. The description covers aspects like the operating system and framework, because of its role in the overall safety and dependability of the whole software system, the research as a precondition for innovative products, and the man-machine interface, which is indispensable for conveying information to the user as well as allowing the user to interact with the robot. The issues that have been faced stem from the hierarchical, layered construction of a complex mechatronic product, where the operation of the machine depends on the smooth cooperation of each layer; In the same way, the overall safety is undermined by the least reliable piece building the system.**

Index Terms **– Mechatronic, Complexity**

I. INTRODUCTION

Mechatronic is a highly interdisciplinary domain, where engineers with different specialization like mechanics, electronics and computer science collaborate with other scientists from more classical domains like mathematics and physics. Because of the sheer amount of people involved, mechatronic product development is extremely difficult. Generally speaking, a mechatronic system consists of several layers (fig. 1), where each layer is able to perform correctly only when the underlying one is behaving correctly, too. The number of interfaces between layers leads to an explosion of the overall complexity of the system, thus rendering the development error prone, and the testing extremely difficult. More precisely, the testing is of outmost challenge because errors are difficult to isolate, as they are often non-repeatable or even due to the wrong interaction of a constellation of layers.

This paper presents the Autonomous Systems Lab's approach to the development of self-contained, autonomous mobile robots. The reliability of mechanical and electronic hardware can be achieved by using industry-standard, off-the-shelves solutions. The wide deployment has already demonstrated (and contributed to) their high degree of dependability, if not *by design* at least *by brute force*.

The choice of the computer science paradigms is by orders of magnitude more complex. The situation is rendered more unfortunate by the fact that most of the mobile robotics research makes use of computer science as the primary tool for realizing itself. We argue that the careful choice of the computer science paradigms, not only eases the development of the proposed innovations, but also fosters them, since the implementor relieved from the computer science issues can better focus on the problem to be solved.

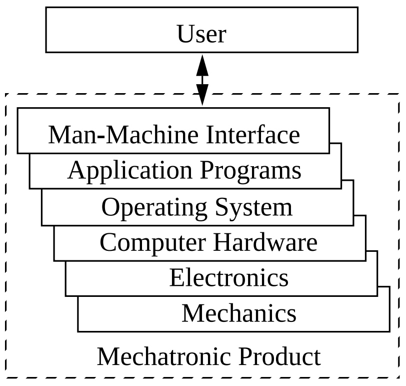


Figure 1: *Abstract view showing the components of a mechatronic system. Note that the performance of each layer relies on the correctness of the underlying layer.*

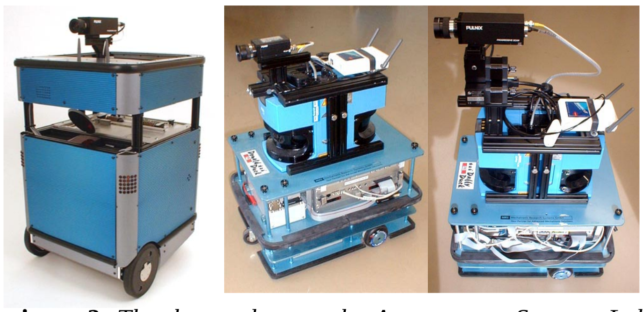
II. THE MOBILE ROBOT: A COMPLEX MECHATRONIC SYSTEM

Mobile robots are still not consumer products. Nevertheless, the success in the research and some innovative products are showing that the time is ripe for popular acceptance. Mobile robots are a great example of the complexity of mechatronic products as they highlight the most dramatic problems and requirements of this domain: Safety, reliability and robustness are the required preconditions from actuators, sensors, electronics, embedded operating system and application programs present in a mechatronic application.

A. Our Autonomous Robots

The Autonomous Systems Lab, EPFL Lausanne, Switzerland, recently built three mobile robots (fig. 2). While they are mainly used for research purposes, they have been crafted by taking into account the application requirements. Because of this, autonomy with respect to perception, energy and processing for fully self-contained autonomous decision-making mobile machines has been addressed in its full complexity already as a research topic.

[†] Autonomous Systems Lab, Swiss Federal Institute of Technology Lausanne (EPFL), CH–1015 Lausanne, (n.tomatis@ieee.org).
[‡] Institute of Robotics, Swiss Federal Institute of Technology Zurich (ETHZ), CH–8092 Zürich, (brega@ifr.mavt.ethz.ch).


igure 2: *The three robots at the Autonomous Systems Lab: Pygmalion, Donald Duck and Daffy Duck.*

B. *Mechanics*

Pygmalion, *Donald* and *Daffy Duck* are built using a differential drive technique. This means that there are two driving wheels and one castor wheel for the balancing. The castor wheel can rotate effortlessly around its own axis. The axis of the two driving wheels are fixed, while the rotational velocity of each wheel can be controlled separately giving a rotation point in the middle of the wheel axis. Furthermore, *Pygmalion* has tactile zones at two different heights, used as emergency bumpers.

C. *Electronics*

The controller consist of a *VME* standard backplane, where a *Motorola MVME 2300* board with a *PowerPC 604ev* microprocessor, clocked at 300 Mhz, is used as the sole processing unit. A VME-based six-axis controller is used for driving the wheels. Among the various peripheral devices, all the robots have three main sensors: the wheel encoders, two *SICK LMS 200* laser range finders, allowing for a 360˚ view, and a *Pulnix CCD* camera. Additionally, *Daffy Duck* has a pan-tilt device upon which a colour CCD camera is mounted. Batteries ensure an autonomy of approximately 4 hours.

III. Operating System and Development Environment: XO/2

The application software is deployed on top of the XO/2 operating system [3]. XO/2 is an object-oriented, hard-real time system software and framework, designed for safety, extensibility and abstraction. It takes care of many common issues faced by programmers of mechatronic products, by hiding general design patterns inside internal mechanisms or by encapsulating them into easy-to-understand abstractions. Careful handling of the safety aspects has been the criterion by which the system has been crafted. These mechanisms, pervasive yet efficient, allow the system to maintain a *deus exmachina* knowledge about the running applications, thus providing higher confidence to the application programmer. The latter, relieved from many computer-science aspects, can better focus his attention to the actual problem to be solved. The following sections will highlight some of the aspects of the XO/2 real-time operating system that are most relevant for deployment in a state-of-the-art mechatronic product.

A. *Automatic Memory Reclamation*

The highly efficient engineering of an application software asks for a highly dynamic, object-oriented, composable software system. Unfortunately, in such an environment the central knowledge of all references that exist for a particular object becomes hard to maintain as the dynamic loading of extensions augments. Even worse, it becomes impossible for a programmer to keep track of references in a safe way when the language doesn't impose restrictions on the passing and copying of references. This brings us to the sheer conclusion that in a dynamically extensible system, explicit deallocation of objects is not feasible.

The only safe possibility for object reclamation is by means of a system-wide mechanism performing automatic storage reclamation: a so-called *garbage collector*. A garbage collector decides upon the liveness of heap objects by their reachability, starting from a working set of global and local references. After complete traversal of the heap data structures, objects that haven't been visited by the collector's marking get disposed.

XO/2 deploys a very robust, real-time compatible *mark-and-sweep* garbage collector with object-finalization that combines good collection performance with no memory requirements at execution time [4]. The latter is more important when the collector is kicked by a low-memory condition, i.e. it can complete the traversal and the collection of the heap-space without demanding memory. Moreover, the proposed solution works very well in a pre-emptive scheduling environment, without blocking nor delaying tasks performing accesses to objects.

B. *Modularisation*

One of the most important design principle is the separation of concerns. This principle requires a system to be structured into subsystems, also called modules. Modules should expose an interface by exporting functions to the clients. The functionality of a module is accessed only by means of its interface; The interface can be generalized enough to hide most of the implementation details, thus establishing and guaranteeing invariants for its states and procedures. Disjoint, orthogonal modules implementing this design principle can be exchanged without invalidating clients, therefore leading to the dynamic composition of the system.

An important precondition for the realization of this design principle is the presence of safe dynamic loading and unloading of compiled units. XO/2 provides the required safety by checking at compile time and at linking-loading time the formal interfaces against the actual ones. Only interface-compatible modules may be loaded in the system, thus posing no threat to the safety of the dynamic composition. The safe unloading of modules is achieved by means of reference counting, lexical scopes and virtual memory ranges, effectively guaranteeing that an entity used by the embedded system will not be unloaded or that stale references will be trapped before execution.

Safe dynamic loading and unloading, along with very short *edit-compile-run* cycles, has been one of the most appreciated features of XO/2. In fact, during the development of a complex application, different programmers can safely test new code modules without undermining the stability of the system and applications.

C. *Process Scheduling*

The principal responsibility of a real-time operating system can be summarized as that of producing correct results while meeting pre-defined deadlines. Therefore, the computational correctness of the system depends on both the logical correctness of the results it produces, and the timing correctness, i.e. the ability to meet the deadlines of its computation.

A real-time application can be modelled as a set of cooperating tasks. These tasks can be classified according to their timing requirements, as hard–real-time, and non–real-time. A hard–real-time task is a task whose timely execution is labelled as critical to the operation of the whole system. Consequently, it is assumed that the missing of the deadline can result in a system failure. Non–real-time tasks are those tasks that exhibit no real-time requirements (e.g. system maintenance tasks running in the background).

Mainstream real-time operating systems implement real-time timing constraints by means of a priority-based, interrupts-based solution. The weakness of this approach relies in the fact that the mapping of the real-time problem from the application space –whose metric is specified in seconds– to the implementation space –whose metric is some artificial value called priority– is left to the application programmer. This approach severely hinders software composition and the deterministic behavior of the software constellation. The first issue stems from the fact that the programmer cannot rely in the third party compliance to some guidelines about priorities. The non-deterministic behavior is a side effect of relying on asynchronous, sometimes external events for deciding the time execution paths.

XO/2's real-time task manager implements a static, earliest-deadline-first scheduling algorithm with admission testing. With this algorithm, the pool of real-time tasks is statically sorted according to their deadlines. The first one, i.e. the one with the shortest deadline, will be set for execution by the scheduler. This task will remain in the foreground, until its normal execution cycle is completed, or when a task characterized by a shorter deadline has been activated by the occurrence of some event, such as the expiration of a waiting period or the user intervention. The process manager is also responsible for dispatching non–real-time tasks, also called threads. Since their computations can be delivered any time, threads are brought to the foreground only when no other real-time task is pending, waiting for being dispatched. The non–real-time scheduler chooses the thread to be scheduled according to its priority. Threads carrying the same priority are taken in the foreground in a round-robin fashion. Anti-starvation mechanisms and priority inheritance guarantee fairness and progress.

D. *Approximation of the Worst-Case Execution Time*

The control of many complex mechatronic products requires for each task the *Worst Case Execution Time* (WCET), which is needed for the scheduler's admission tests and subsequently limits a task's execution time during operation. If a task exceeds the WCET, this situation is detected and either a handler is invoked or control is transferred to a human operator. Such control systems usually support pre-emptive multitasking, and if an object-oriented programming language (e.g., Java, Oberon) is used, then the system may also provide dynamic loading and unloading of software components.

Only modern, state-of-the art microprocessors can provide the necessary computation cycles, but this combination of features (preemption, dynamic un/loading of modules, advanced processors) creates unique challenges when estimating the WCET. Preemption makes it difficult to take the state of the caches and pipelines into account when determining the WCET, yet for modern processors, a WCET based on worst-case assumptions about caches and pipelines is too large to be useful, especially for big and complex real-time products. Since modules can be loaded and unloaded, each task must be analyzed in isolation, without explicit reference to other tasks that may execute concurrently.

To obtain a realistic estimate of a task's execution time, XO/2 uses static analysis of the source code combined with information about the task's runtime behavior. Runtime information is gathered by the performance monitor that is included in the processor's hardware implementation. The predictor [7] is able to compute a good estimation of the WCET even for complex tasks that contain a lot of dynamic cache usage, and its requirements are met by today's performance monitoring hardware.

IV. Research

The mechatronic research represents a valid starting point for innovative applications and products only when researchers address fundamental questions by taking into account application relevant limitations and problems. Our research focuses on fundamental questions about sensor modelling, localization, map building, obstacle avoidance, motion control and man-machine interaction. The next sections will present some results achieved in these domains.

A. *Sensors and Modelling*

The mobile robots have three main sensors: the wheel encoders, a 360˚ laser range finder and a CCD camera. In order to optimise the perception, the first step that has to be taken centres around the modelling. Measurements are –intrinsically– uncertain. By knowing the physical characteristics of the applied sensors, uncertainties can be isolated, modelled and propagated up to the application level. This allows us to combine the use of probability theory to represent the uncertainty of the geometric elements of the environment and of the robot as well. Moreover, probabilistic position estimators have turned out to be much more reliable.

1) *Wheel Encoders (Odometry)*

Non-systematic odometry errors occur in two spaces: the joint space and the Cartesian space. Effects of wheel slippage, uneven ground and limited encoder resolution appear in the joint space. They are modelled by means of the physically well-grounded model presented in [6].

Effects of external forces (mainly collisions) occur in the Cartesian space. However, they are even more difficult to model because of their unpredictablity. This is the reason why they are not taken into account in our approach.

2) *Laser Range Finder*

In this case the constant uncertainty given by the supplier of the sensor has been adopted.

3) *Camera*

The camera system is calibrated by combining the method described in [11] with the spatial knowledge from a test field. This provides a coherent set of extrinsic, intrinsic and distortion parameters. Uncertainties from the test field geometry as well as those caused by noise in the camera and acquisition electronics are propagated through the camera calibration procedure onto the level of the camera parameters.

B. *Multisensor Metric Localization*

Having a multisensor setup allows for the use of different features with complementary characteristics, leading to a consequent growth in robustness and precision. This section addresses the question of how to combine data from different sensors.

1) *Environment Representation*

Features are infinite horizontal lines for the laser range finder and vertical edges for the vision system. The a priori map contains 191 infinite lines and 172 vertical edges for the $50 \times 30\,m$ portion of the institute building shown in fig. 3. This environment model distinguishes itself by its extreme compactness, having a memory requirement of about 30 bytes $/m^2$.

Laser Range Finder: The algorithm for line extraction has already been used in [1]. It performs three main steps:

- Model independent segmentation on groups of points.
- Nonlinear regression fitting.
- Association by means of clustering with Mahalanobis distance matrix.

The method delivers lines and segments with their first order covariance estimate using polar coordinates.

Camera: Vertical lines are extracted in four steps:

- Vertical edge enhancement with a specialized Sobel filter.
- Non-maxima suppression using dynamic thresholding.
- Edge image calibration: The horizontal position of each edge pixel is corrected.
- Line fitting reduces to a one-dimensional problem for each image column.

Uncertainty from the camera electronics is modelled on the level of the uncalibrated edge image. Together with the uncertainty of the calibration parameters it is propagated through calibration and line fit, yielding the first two moments of the vertical edges.

2) *Enhanced Kalman Filter (EKF) Navigation*

For metric localization an extended Kalman filter is used. A localization cycle consists of five steps [5], [8]:

State prediction: The state and its associated covariance are determined from the odometry based on the previous first and second state moments.

Observation: The parameters of the extracted feature constitute the vector of observations and its associated observation covariance matrix.

Measurement prediction: The modeled features in the map get transformed into the frame of the observations. The first moments are computed by the global-to-local transform. Error propagation is done by a first-order approximation which requires the Jacobian with respect to the state prediction.

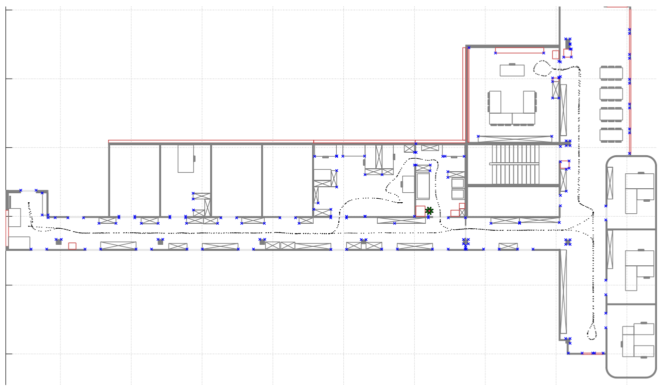
Matching: Since the Kalman filter represents and propagates a single Gaussian distribution for the robot pose, false pairings can lead to irreversible filter divergence, i.e. a lost-situation asking for manual intervention. The matching step is therefore of high importance. Because of this the laser observations are integrated first, since they typically exhibit far better mutual discriminance, thus making their matching less error-prone. These are followed by the vertical edges from the camera, where ambiguous matching situations often occur.

Estimation: Successfully matched observation and predictions are then used to compute the a-posteriori estimates of the robot pose and associated covariance.

3) *Results*

The goal of this approach was, on the one hand, to have a valid EKF implementation for our robots and, on the other hand, to compare the laser-only configuration against the laser and vision setup. For this comparison *Pygmalion* ran the trajectory shown in fig. 3 ten times, five with the laser-only configuration and five with laser and vision [1].

The results are summarized in table 1. The error bounds show that, based on the uncertainty model, the robot is with a 95% probability within twice this value. The vision information reduces this uncertainty in x and y in equal measure (-20%), but particularly in the orientation (-40%). This holds, even if the number of matched vertical edges is moderate.



igure 3: *The 140m trajectory. Pygmalion ran five runs with laser-only localization and five runs with laser and vision.*

	Laser	Laser and vision
$2\bar{\sigma}_x$	1.31 cm	1.07 cm
$2\bar{\sigma}_y$	1.35 cm	1.05 cm
$2\bar{\sigma}_\theta$	0.92°	0.56°
$\bar{n}_l / \bar{n}_v$	2.73 / –	2.66 / 2.00
$\overline{t_{exe}}$	64 ms	411 ms

Table 1: *Overall mean values of the error bounds, the number of matched lines n_l and matched vertical edges n_v, and the average localization cycle time.*

C. Obstacle Avoidance and Motion Planning

Obstacle avoidance in mobile robotics is the problem of moving from a starting point to a given goal with an arbitrary constellation of obstacles. For this, the approach has to take the kinematic and sensing capabilities of the robots into account.

The algorithm we devised has been split into two independent modules: the first one implementing local obstacle avoidance and the second one implementing a path planner which generates a path of points from the start to the goal [10]. The local obstacle avoidance is based on the Dynamic Window approach. The safe robot behavior is ensured by selecting only motion commands that allow the robot to come to a halt before collision. Above the dynamic window there is a path planner which gives intermediate goal points to the local obstacle avoidance. The path planner makes use of the distance transform which is a grid-based wave propagation technique: The raw laser scan points are put into a grid, a wave propagates from the goal point until all points in the grid have been reached. A path from the start to the goal is eventually found by following the negative gradient from the start.

D. Man-Machine Interaction

Man-machine interaction is the layer closest to the end-user and therefore the information presented has to be suitable for human beings. Since interaction is the bi-directional exchange of information, this layer has to service input devices as well as output channels (fig. 4).

1) Physical Interaction

For the physical interaction the CCD camera is used as an input device locating human beings in the environment. The robot detects human motion using statistical change detection algorithms. Output devices are the pan-tilt head being directed towards the chosen user, which is addressed by the robot with synthesized speech. Simple tracking algorithms allow the robot to interact with the person even when he is moving.

2) Web-Interfacing

Testing algorithms like localization and obstacle avoidance on an autonomous self-contained robot requires a means for the researcher to check the algorithmic reactions to the machine's perception of the world. However the perceived data and the processing results remain embedded in the mobile vehicle until they are explicitly transferred to a local PC for analysis. This can be done by tracing the robot position (odometry), by saving all the raw data from the sensors, the extracted features and the results of each algorithm, then by transferring this information when an experiment is finished and analysis is performed off-board. Nevertheless, this procedure has several disadvantages mainly due to the lack of correspondence between the behaviour of the robot and its data. On-line supervision (as web-interface in fig. 5) is therefore not an option, it is instead an important tool for speeding up the advances in applications such as mobile robotics research by allowing on-line detection of characteristics of the tested approaches [9].

V. The Application: Remote Exploration

The "Computer" trade show is an annual fair for computer hard- and software at the *Palais de Beaulieu Exposition Centre* in Lausanne, Switzerland. Our laboratory was present during the four days –from May 2nd to May 5th 2000– giving visitors the opportunity to control *Pygmalion* by means of web-interfacing (fig. 5), while the robot itself was at EPFL.

The Computer 2000 event was the final testbed for our metric localization system, where we were mainly interested in long-term reliability under application-like conditions [2]. Furthermore, it was a great way to test our approaches in physical man-machine interaction (fig. 4) and web-based interfaces (fig. 5). The setup was active during normal office hours with an average of about 7 hours up-time per day. The environment exhibited typical dynamics from people, doors, chairs and other robots, as well as daylight illumination. Several doors were open into the corridor, thus limiting the space available for robots maneuvers. Travel speed has been limited to 0.4 m/s since the robot had to share its environment with persons, some of them not engaged into robotics. The obstacle avoidance was active during the event. The web-interface (fig. 5) allowed to give navigation commands (e.g. go to office) to the robot.

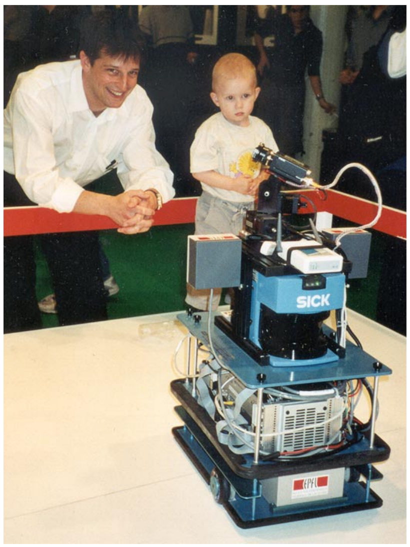


Figure 4: *Daffy, after detecting a young visitor, interacts with him by speaking, tracking him and moving the pan-tilt head for mimicking expressions. It detects human movement by means of the CCD camera and closes the interaction loop with its speech synthesizer and gestures as a combination of robot and pan-tilt movement.*

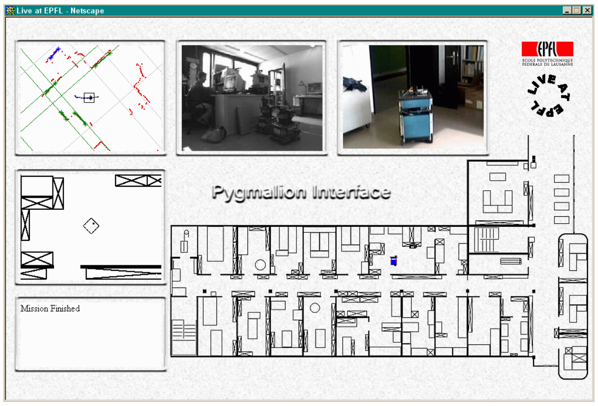


***Figure 5:** The Pygmalion web-interface, a plug-in-free Netscape application. It provides context-sensitive menus on the map and all subwindows with intuitive click-and-move-there commands for robot teleoperation. Feedback is given by means of images of the raw and localization data from the laser range finder (top-left), an embarked camera (top middle), an external web-cam (top-right) and the robot animated in its model map (left middle).*

A. Results

The event statistics of Computer 2000 are shown in table 2. A lost situation is defined as a mission whose goal could not be achieved due to a localization error and which required manual intervention in order to re-localize the robot. Missions where the robot went lost due to a collision with an invisible object (e.g. glass door or object lower than the beam height of the scanner) and where the robot was already lost (after such a collision) have not been considered.

When many people are around the robot or when the odometry delivers inconsistent estimates due to uneven floors, it can happen that there are no matched features during a certain period. We counted 14 out of 724 missions where the robot had no matches during 10 seconds, and 21 missions where it had no matches during 5 seconds. None of them required manual intervention during or after the mission.

We had therefore no loss situations due to localization. However, in about ten cases the robot required human intervention due to collisions with “ghost” objects.

Feedback from visitors about man-machine interaction was very positive: Daffy (fig. 4) was the main attraction of our stand and the interface turned out to be very user-friendly.

Hours of operation	28
Environment size	50 x 30 m
Environment type	office, unmodified
Overall travel distance	5,013 m
Average speed	0.2 m/sec
Number of missions	724
Number of localization cycles	145,433
Number of lost-situation	0
Number of unknown collisions	~ 10

***Table 2:** Overall statistics for the Computer 2000 event.*

VI. Conclusions

Mobile robots are complex mechatronic products that emphasize requirements like safety, reliability and robustness. The very same requirements define the overall performance of the robot by posing threats to its run-time safety. They act as preconditions for all the robot's components: Actuators, sensors, electronics, embedded operating system and application programs have all to show the same quality, since the overall safety is undermined by the least reliable piece building the system.

This paper presented the Autonomous Systems Lab’s approach to the development of self-contained, autonomous mobile robots, their challenges and the devised solutions. Backed up by the experience we collected, we argue that progress in mobile robotics requires the researchers to access and improve all modules that makes up the system, from low-level mechanical components to high-level reasoning systems. Moreover, besides the choice of the hardware (often straightforward), the computer science paradigms are crucial, since they play a primary role during the development of the intelligent mechatronic products. XO/2, an object-oriented, hard-real time operating system has been adopted, because of its support for safety, extensibility and abstraction.

Research has been described as an important step to innovative products. The results of our research culminated in a real application. The remote exploration application has been presented at the “Computer 2000” trade show in Lausanne, Switzerland, where the visitors were able to control *Pygmalion* by means of a web-interface.

References

[1] Arras, K., N. Tomatis, et al. (2000). Multisensor On-the-Fly Localization Using Laser and Vision. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000), Takamatsu, Japan.

[2] Arras, K. O., N. Tomatis, et al. (2000). “Multisensor On-the-Fly Localization: Precision and Reliability for Applications”. To appear in Journal of Robotics and Autonomous Systems.

[3] Brega R. (1998). A real-time operating system designed for predictability and run-time safety. In Proc. of The Fourth Int. Conf. on Motion and Vibration Control (MOVIC), Zurich.

[4] Brega R., G. Rivera. (2000). Dynamic Memory Management with Garbage Collection for Embedded Applications. USENIX Workshop on Industrial Experiences with Systems Software (WIESS 2000), San Diego, CA.

[5] Bar-Shalom Y., T.E. Fortmann. (1988). “Tracking and Data Association”, Mathematics in Science and Engineering, Vol. 179, Academic Press Inc., 1988.

[6] Chong, K.S. and L. Kleeman (1997). Accurate Odometry and Error Modelling for a Mobile Robot, IEEE International Conference on Robotics and Automation, NM, USA.

[7] Corti, M., R. Brega. et al. (2000). Approximation of Worst-Case Execution Time for Preemptive Multitasking Systems. ACM SIGPLAN LCTES'2000 (Workshop on Languages, Compilers, and Tools for Embedded Systems), Vancouver B. C., Canada.

[8] Crowley, J.L. (1989). World Modeling and Position Estimation for a Mobile Robot Using Ultrasonic Ranging, IEEE International Conference on Robotics and Automation, Scottsdale, AZ.

[9] Moreau, B., N. Tomatis et al. (2000). Multimodal Web Interface for Tasks Supervision and Specification. Proceedings of SPIE Vol. 4195: Telemanipulator and Telepresence VII. Boston, USA.

[10] Persson, J. (2000). Obstacle Avoidance for Mobile Robotics. Diploma thesis, Dept. of Electrical Eng., Linköpings University, Linköpings, Sweden.

[11] Prescott, B., G. F. McLean (1997). Line-Based Correction of Radial Lens Distortion. Graphical Models and Image Processing. 59(1).