

# THE NTUA SPACE ROBOTIC EMULATOR: DESIGN AND EXPERIMENTS

**Evangelos Papadopoulos, Iosif S. Paraskevas, Georgios Rekleitis, and Thaleia Flessa**

National Technical University of Athens  
Mechanical Engineering Department - Control Systems Laboratory  
9 Heroon Polytechniou Str., 15780 Athens, Greece  
Email: egpapado@central.ntua.gr

## ABSTRACT

The importance of space robots in satellite servicing, in EVA assistance, in removing orbital debris and in space exploration is growing, driven by space commercialization and exploration efforts. A successful deployment of such robots requires analytical and experimental task validation. With this aim, we present the main characteristics of the software and hardware space robot emulator developed at the NTUA.

## 1. INTRODUCTION

The commercialization of space, the growing number of structures in orbit, and the need for autonomous robotic exploration and servicing in the form of orbital and planetary agents as a key step in space exploration, all require robotic systems capable of fulfilling tasks such as construction, maintenance, assistance, docking and inspection. Such robotic systems are expensive to build and present numerous difficulties before they are considered for launching. Technology demonstration missions, such as the ETS-VII, the Orbital Express and the DART, while called “demonstration missions,” are usually expected to succeed at least partially. Such demonstrators are designed with minimum fault tolerance, therefore allowing less flexibility. JAXA’s ETS-VII, is a successful example of autonomous robotic servicing. During this mission, a rendezvous and docking procedure was successfully completed [1]. In addition, visual inspections, an ORU exchange and a liquid transfer operation to and from an ORU were also tried. DARPA’s Orbital Express successfully demonstrated rendezvous manoeuvres between a target and a chaser, autonomous capture of a free-flying target and ORU replacement with the help of a robotic manipulator [2-4]. NASA’s DART mission attempted to demonstrate autonomous rendezvous between a spacecraft and a satellite and the subsequent capture of the satellite [5]. However the mission was aborted during the approach phase because of a spacecraft/ satellite collision. A survey of robotic systems with servicing in-space capabilities such as assembly, repair and maintenance is given in [6].

A space-like environment can be simulated using either software or hardware simulators such as neutral buoyancy facilities, gravity compensation mechanisms, parabolic flights and planar or rotational systems using air-bearings. Water tanks and other neutrally buoyant systems have the advantage that they allow for simulating space motion in 3D. However, the existence of water resistance hampers the realism of the simulation, thus making this approach more suitable for training astronauts in zero gravity slow motions. Another approach is the use of a mechanism that supports the robotic system and negates the effects of gravity, [7], [8]. This approach, although promising, has yet to overcome the problems due to mechanism singularities and imperfections, and gravity loading of system joint actuators. Parabolic flights allow for experiments in microgravity but they are severely limited by the cost and time available for the experiments. A different approach is a planar simulator, which negates the effect of gravity by employing a practically frictionless motion of the simulated robotic system on a horizontal plane. This motion can be achieved by several methods, such as by air bearings. Although they can emulate a zero gravity environment accurately, planar emulators have the disadvantage of not being able to emulate 3D effects. Despite this, planar simulators using air-bearings are perhaps the most versatile and less expensive in comparison to other methods, and allow for repeated and thorough testing of control algorithms, verification of dynamics, rendezvous manoeuvring and construction techniques. A review of air-bearing spacecraft simulators is presented in [9], which clearly illustrates their versatility and reliability.

Test facilities simulating a space-like environment have been developed among others in the USA (MIT, Stanford, NASA), Europe (University of Southampton, University of Padova) and Japan (Tokyo Institute of Technology, Tokyo University). The MIT’s SPHERES project is a testbed designed for testing formation flying algorithms and consists of three small satellites, each attached to a puck that hovers over a glass table using air-bearings, and moves using thrusters [10]. The SPHERES project was flown in ISS in 2007, where formation flying experiments were conducted [11]. Stanford University’s Aerospace Robotics Laboratory also has a planar simulator where three active free-flying vehicles and one passive target vehicle hover over a granite table using air-bearings. The active vehicles are propelled by firing thrusters. Among others, the purpose of the simulator is to test formation flying control, assembly and construction methods [12, 13]. The University of Padova, in collaboration with the “G. Colombo” Center of Studies and Activities

for Space (CISAS), has developed a robot with an anthropomorphic manipulator that hovers over a small table using air-bearings and moves using thrusters [14]. The University of Southampton has a simulator that consists of a glass table on which mock-up satellites can glide. Each mock-up consists of a base disk that glides on the table, and a satellite frame attached to the base that can rotate [15]. At Tokyo Institute of Technology, an experimental system has been developed with the aim of testing the dynamics of space robotic manipulation. The simulator consists of a 6-DOF manipulator that hovers using air-bearings over a flat floor. Also, Tokyo University has developed a small robot, tethered to a service satellite, with the aim of releasing the robot to service and capture satellites and then return to the service satellite. The concept is tested at a planar simulator where the robot hovers over a metal table using air-bearings and is tethered to a mock-up of the service satellite, the service satellite being outside the table [16].

This paper presents the software and hardware space robot simulator developed at the National Technical University of Athens (NTUA). This consists of a software simulator with an animated graphic representation and a planar hardware emulator. The software simulator allows for a 3D parameterization of a robot consisting of a base and several appendages, derivation and simulation of the dynamic equations, including control algorithms, and visualization of the results using OpenGL libraries. The hardware planar emulator consists of a granite table of minimum roughness and a small robot supported on three air-bearings. The robot is capable of horizontal frictionless motion on the table, thus allowing for 2D emulation zero gravity in a laboratory environment. The robot is fully autonomous. Its propulsion autonomy is achieved by an on-board CO<sub>2</sub> tank used to provide gas to the air bearings and to three couples of propulsion thrusters. The robot is also equipped with a reaction wheel to control the robot's angular momentum. The computational autonomy is achieved with a PC104 mounted on the robot. Power autonomy is achieved with a set of on-board batteries. The novelty of this configuration is that the robot is not only of low mass and completely self-contained but also it is composed of subsystems similar to those of a space system, therefore making the emulator significantly more realistic. The NTUA simulator provides a low-cost, long duration, and easily reconfigurable platform that allows for the analytical and experimental validation of different control, dynamics, and planning schemes, thus facilitating the transition from theory and analysis to application.

## 2. SOFTWARE SIMULATOR

The software simulator for space robots on orbit consists of three basic components. These are (a) the dynamic equations of motion, (b) the numerical simulation including various control algorithms and, (c) the animated graphic representation. The equations of motion are obtained using the mathematical package Mathematica<sup>®</sup>, while the system's behaviour under the chosen control method is simulated using the simulation package Simulink<sup>®</sup>. The animated graphic representation is realized in a program developed in our lab at the NTUA, and is based on OpenGL libraries. This program uses the Simulink<sup>®</sup> model data output, and produces the corresponding animation.

(a) *Dynamics and Simulation*: Orbital robotic systems consist of a base with several appendages, such as manipulators, communication antennae etc. Such a system in 3D has at least six degrees of freedom for the base position and orientation, plus as many as the number of the joint variables of the appendages. Thus, the user of the simulator must be able to define, not only the inertial and geometric properties of the base, but also those of the appendages, as well as their number and the location of their supports.

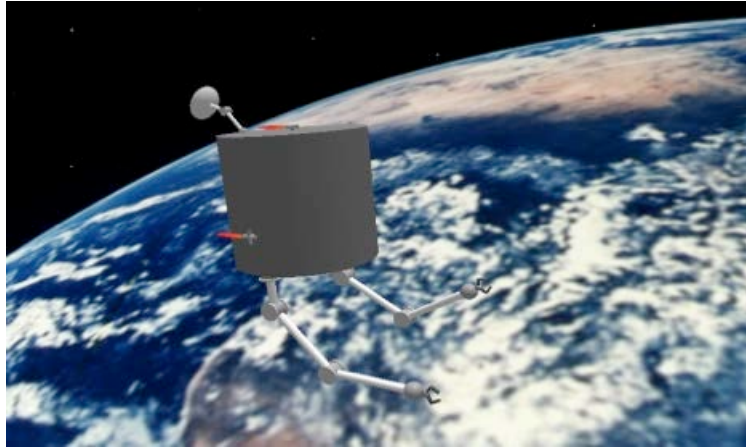
To begin the dynamic modelling of such a system, a set of generalized coordinates  $q_i$  is chosen, describing the position and orientation of the base, as well as the position of the links of the manipulators. In order to obtain the dynamic equations of motion, the Lagrange formulation is used. Note that the systems' total potential energy is equal to zero for orbital systems. Using Lagrange's equations of motion for each generalized coordinate, the final form of the systems' equations of motion, is obtained as

$$\mathbf{H}\ddot{\mathbf{q}} + \mathbf{C} = \mathbf{Q} \quad (1)$$

In (1),  $\mathbf{q}$  is the  $k \times 1$  vector of the generalized coordinates,  $\mathbf{H}$  is a  $k \times k$  matrix related to the inertia properties of the system,  $\mathbf{C}$  is a  $k \times 1$  vector containing all nonlinear velocity terms and  $\mathbf{Q}$  is the  $k \times 1$  vector of the generalized forces. Matrices  $\mathbf{H}$  and  $\mathbf{C}$  are obtained symbolically using Mathematica<sup>®</sup>. The code that generates these matrices can be easily modified to account for the number of manipulators and their links of the simulated robotic system. The inertia and geometric properties of the system are introduced as code parameters, and remain as parameters that can be changed in Simulink<sup>®</sup>, providing an easy development of what-if scenarios, as long as the structural description of the robotic system remains the same. If any of these parameters is changed, new  $\mathbf{H}$  and  $\mathbf{C}$  matrices are obtained automatically.

(b) *Graphic Animation*: In order to visualize the results of the simulations, an animation program was developed. Using this program, the user can have a better view of the response of the simulated systems. The program runs in Windows and has two parts, one creating a space robot and one for animating its motion, according to simulation data.

The idea is to create each robot with the minimum possible number of input parameters. The program uses these to define the robot using simple shapes and surfaces. The input parameters concern the base of the robot, which can be a rectangle, a cylinder or an orthogonal regular prism, the manipulators, the thrusters used for moving in space and the antenna(s) used for communicating with earth and other space robots. In addition, the number of manipulators and the position and orientation of their base coordinate systems are given to the program. The appendages are drawn using the Denavit - Hartenberg parameters. The thrusters are entered by their position and orientation. Fig. 1 shows a space robot with a cylindrical base, twelve thrusters and two manipulators with four joints each, created using the above steps. Note that each robot is created once and then can be used in different simulations.



**Figure 1.** A two manipulator, twelve thruster space robot moving on orbit.

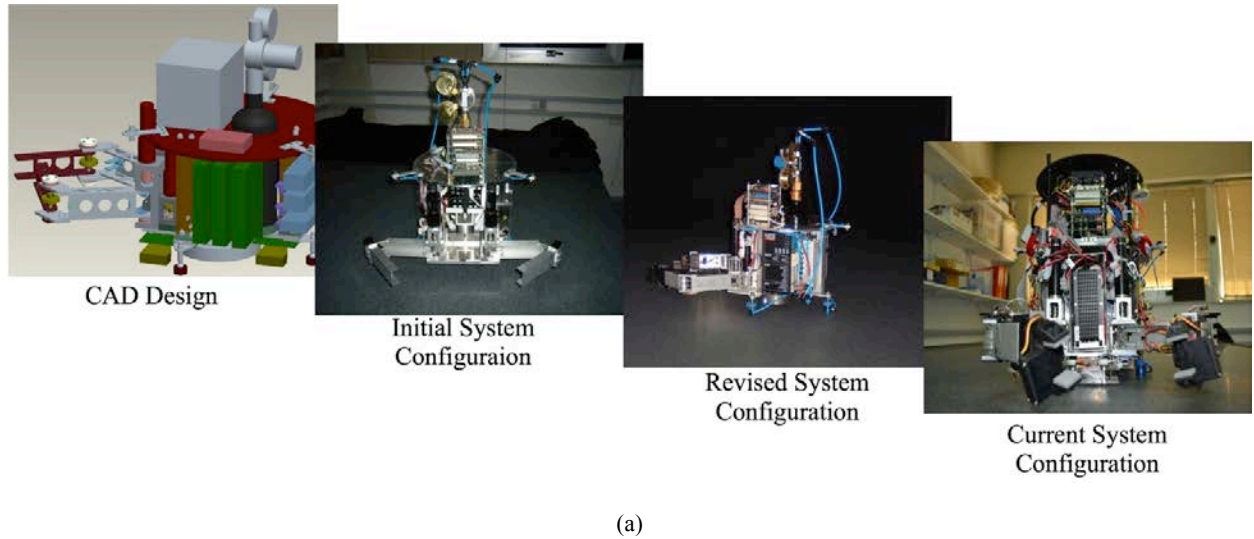
The time interval between two frames in the animation program is 50 ms, so a fixed step of 50ms is used in the Simulink® integrator providing the response data. The total time of the animation can last for 500 s, resulting in 10,000 frames. The frames can show the view from a fixed camera or from one that moves with it, as desired. Other options offered by the program include the ability to draw the path of the robot and the capability for three additional cameras.

### 3. HARDWARE SIMULATOR

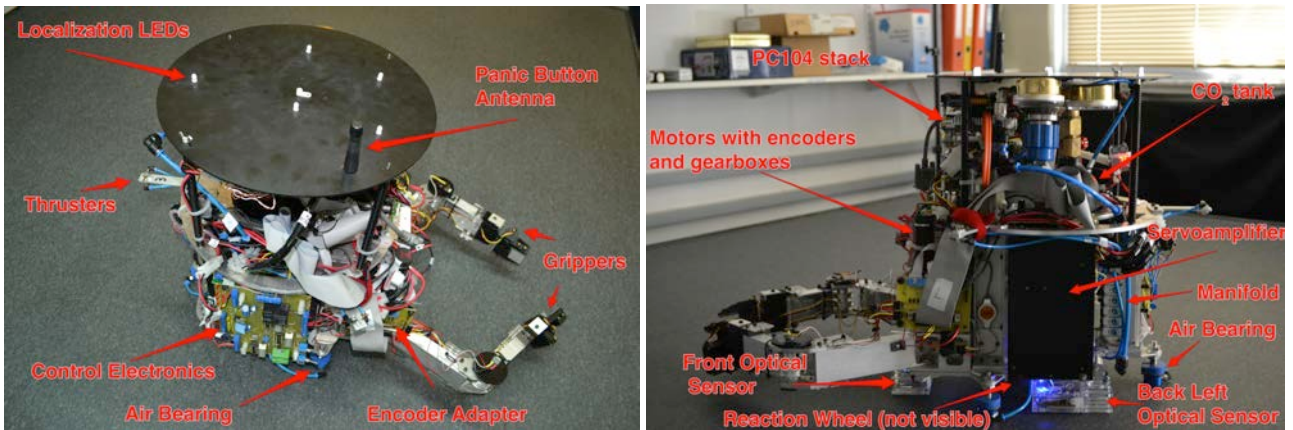
At the NTUA, an experimental testbed was developed for motion studies in zero gravity. The design specifications for the testbed included: (a) Realistic emulation of a zero-g environment, (b) A design that can fit into the limited space of our laboratory, (c) Lightweight design and (d) Autonomy in terms of energy, mobility and computational power.

After consideration of alternative candidates, a planar simulator on a flat surface was chosen, as the best in terms of fidelity and development feasibility. Various space robots or other components can float freely on the flat surface and negate gravity, as its vector is perpendicular to the motion and the applied forces. During the years from the conceptual design to the current configuration, the robot design has undergone through various alterations, see Fig. 2(a). The surface where the robot floats is presented in Fig. 2(b).

For emulation of the zero-g 2D environment, gas under pressure is used, which flows through a number of air bearings. A 10 micron gas film between the floating system and the flat surface is created, that develops a force normal to the surface, lifting the air bearing-mounted robot and canceling the gravity force. The robotic mechanism is mounted on top of the air bearings and carries a gas tank, eliminating external forces or torques that would exist if hoses were used instead. The system is mounted on 3 air bearings and has a dry weight of about 16.5 kg, which is well within the air-bearings required load. The weight of the necessary CO<sub>2</sub> increases the total weight (wet weight) to about 17 kg. Looking at the alternative gases, air and CO<sub>2</sub> were the most promising, as they are readily available and harmless. However, liquid air would be available at 250 bar, which can be a dangerous pressure, while CO<sub>2</sub> needs only 60 bar. Power autonomy is achieved using Li-Po batteries divided into two isolated circuits with common ground in order to avoid unwanted ground loops: one low-voltage circuit for the computer and electronics (nominal voltage 5V) powered from a PC104 DC/DC Converter, and one high-voltage circuit for the servomotors and valves (nominal voltage 24V) powered by one switching DC/DC Converter. The main elements of the robot are presented in Fig. 3.



**Figure 2.** (a) Evolution of the robot from concept to current configuration and (b) the flat granite table.



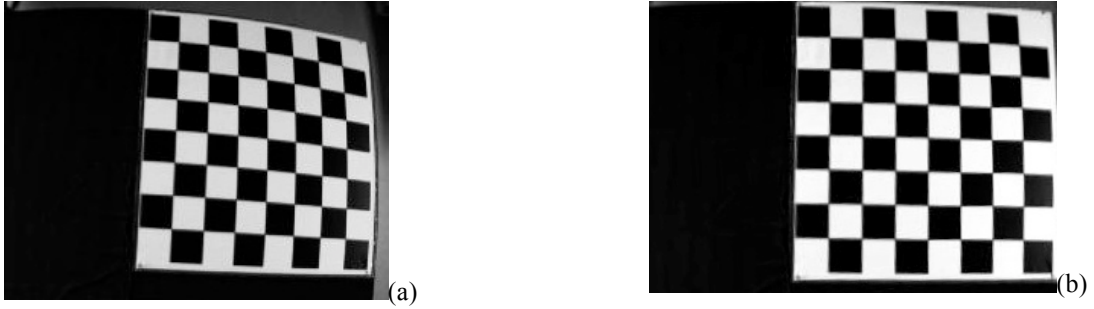
**Figure 3.** Main elements of the space robot in two views.

The flatness of the surface had to be guaranteed both microscopically and macroscopically. The former is important since the local roughness of the surface must be significantly less than the width of the gas film, while the latter is important to avoid a wavy surface that could render the floating robot unstable. A three-tone, special finishing granite table, with maximum anomaly height of  $5\mu\text{m}$  was chosen. The table has a surface area of  $2.2 \times 1.8 \text{ m}^2$ , standing on six adjustable height legs. The table deflection is negligible and its tilt after leg adjustment is less than  $0.01^\circ$ . The table has been placed in our basement lab, where measured environmental vibrations were minimal.

## Sensors

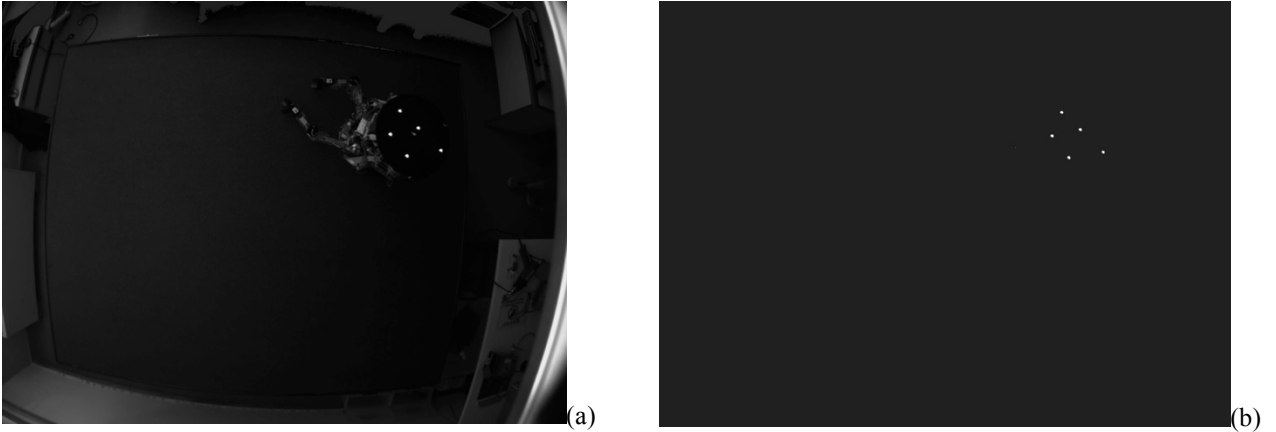
(a) *Camera*: An overhead camera is located above the field of action of the robot to track its position and orientation. To this end, a number of (at least three) LEDs are mounted at the corners of a known polygon, on top of the robot, and are tracked by the camera. Images are sent to an off-board computer, where a process determines robot position and orientation via image processing. This information is sent wirelessly via TCP/IP to the robot's processing unit (PC104).

The camera and its lens were chosen so that the whole action area is contained within the image, with satisfactory resolution. The calibration of the camera was performed off-line, using special two-dimensional known images. The parameters defined after the end of the calibration process, are the *intrinsic* (including the focal length in pixels, principal point coordinates, skew coefficient defining the angle between the x and y pixel axes and radial and tangential distortion coefficients) and the *extrinsic* (including the rotation and translation matrix). The calibration technique used employs a two-dimensional pattern at various unknown orientations and was proposed by Z.Y. Zhang [17]. Thirty photographs of a chess pattern have been taken at different positions and orientations, covering the whole action area. The camera calibration process was also used to reduce the distortions from the use of the fisheye lens, Fig. 4, [18].



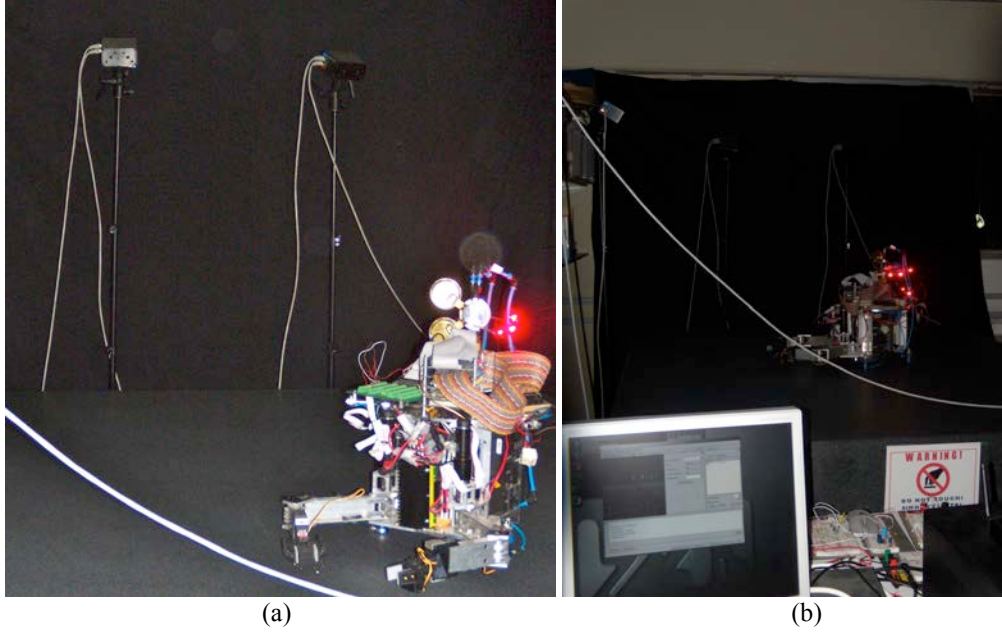
**Figure 4.** Photos of a chess pattern. (a) before and (b) after camera calibration.

In order to be able to send the pose of the robot to the PC104, one process for image capture and two tasks for image processing and data transfer have been developed in a C program that continuously captures images, processes them to obtain robot pose and sends these data to the robot's PC104. At every iteration, the image capturing is performed using standard libraries and functions in C, provided by the manufacturer of the camera. Then, after the image is obtained, halftoning is used in order to isolate the pixels that correspond to the LED's, while turning the rest of the image to black, as can be seen in Fig. 5.



**Figure 5.** Frame captured by the camera: (a) before and (b) after half-toning to reveal only LEDs.

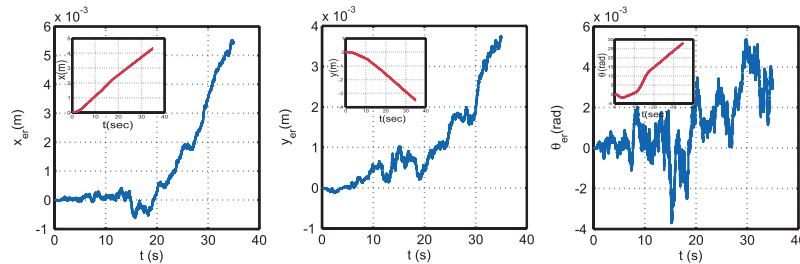
The coordinates of the centers of the LEDs are then calculated and the parameters obtained during the calibration process are used in order to alleviate the distortion of the image. The undistorted LED centers coordinates are then used to obtain the robot pose by means of a theoretical approach developed at the NTUA that will be presented at Sec. 6. The obtained pose of the robot is then transferred to the robots' PC104 via wireless TCP/IP. Note that, the image processing and pose calculations are performed at the off-board computer, in order not to burden the robot PC104 with the extra calculation effort. The final evaluation of the motion data will be performed using the PhaseSpace motion capture system of our lab, see Fig. 6.



**Figure 6.** (a) Robot with Phasespace LEDs on the granite table and Phasespace cameras (b) interaction PC with robot and Phasespace system on the background.

(b) *Optical Sensors:* These are used as alternative means for calculating robot position and orientation. The sensors use optical flow techniques; by comparing successive photos of the surface, they measure the differential displacement at each sampling instant versus the position at the previous instant, and provide a pair of  $dx$  and  $dy$ . Two sensors with known distance among them, produce therefore four values per instant, and the three unknown parameters ( $x$ ,  $y$ , and  $\theta$ ) can be calculated geometrically. Optical sensors have the advantages of having a very high sampling rate, compact size, great accuracy and low cost. Their disadvantage is that the error is accumulated over the time.

This feedback system uses laser lighting, thus enhancing the quality of measurements, and its resolution can exceed 2000 cpi. The error is proportional to speed, but even at high speeds (higher than the actual speed of the robot) it was measured to be less than 5 mm over a displacement of 2 m, see Fig. 7. Using various techniques (e.g. Kalman Filtering) and information from the camera (which has lower frequency compared to the optical sensors), the results are filtered and/or corrected upon predetermined intervals. For redundancy, three of these sensors are being used, producing in total six values per instance. Currently 3 PCBs from small Off-The-Shelf computer mice have been installed. Their cpi is about 1600 cpi and they are located in custom made supports. There is the capability of easily replacing any sensor with a different one (in terms of design and/or specifications). The program to control the received data and determine the position and the attitude of the robot is under debugging. Soon after there will be a calibration procedure using again the CSL's Phasespace system. Again the theoretical approach developed at the NTUA and presented at Sec. 6 will be used.



**Figure 7.** Simulation using 3 optical sensors and least squares techniques for obtaining displacements (inserts) from filtered measurements. Graphs show the accumulated error in the  $x$ ,  $y$ -axes and rotation about the  $z$ -axis.

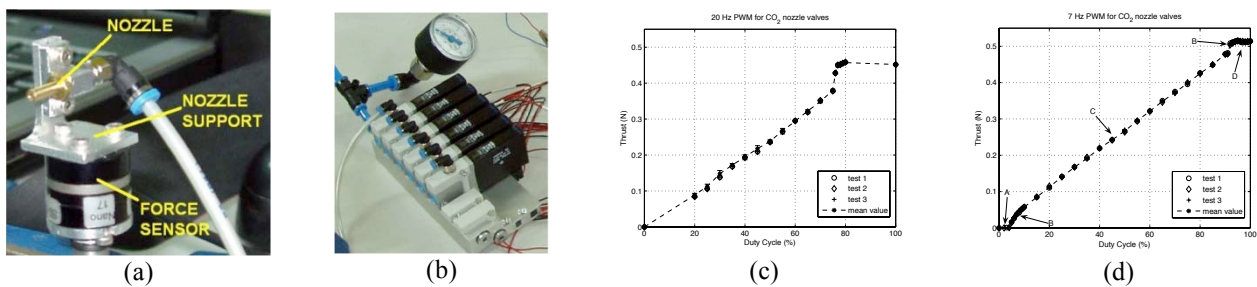
(c) *Other Sensors:* A number of other sensors are being used also. These include force, Hall, and voltage sensors and encoders. Force sensors are used to measure gripper-applied forces. Hall sensors on the manipulators signal to the PC104 the reach of predefined angle positions. This is useful mainly during initialization, where the telemetry can

identify starting positions. Voltage meters, and other proprioceptive sensors on the motherboard inform the user, a PIC microcontroller, and the PC104 of possible malfunctions, in order for emergency actions to be performed. Additionally, each motor is equipped with digital encoders that send joint angular positions to the PC104.

## Actuators

(a) *Thrusters*: The propulsion autonomy is achieved by the thruster system, Fig. 3, which uses CO<sub>2</sub>, and a number of regulators that reduce the tank pressure to less than 7 bar. The electronic circuitry controls the gas flow using Pulse Width Modulation (PWM) and actuates 2-way on-off solenoid valves. This circuit receives 0-5 V signal in PWM format from the PC104 and outputs a PWM high power voltage at 0-24V. The use of PWM allows values of thrust in a continuous range, using the technology of on-off thrusters, which are common for actual space applications.

One key parameter was determination of the value of thrust we can obtain from each thruster. Using a force sensor, see Fig. 8(a), and actuating the solenoid valves under various frequencies (5-50Hz) and duty cycles (0-100%), the optimal envelope has been defined. A common inference from the experimental data was a dead band (signal too slow to overcome internal friction) and a saturation region (signal faster than the time constant of the valve). Higher frequencies resulted in wider dead band and saturation regions, see Fig. 8(c, d). The 7 Hz PWM signal has been chosen as the optimal one, resulting in an effective duty cycle lies between 10-90%.

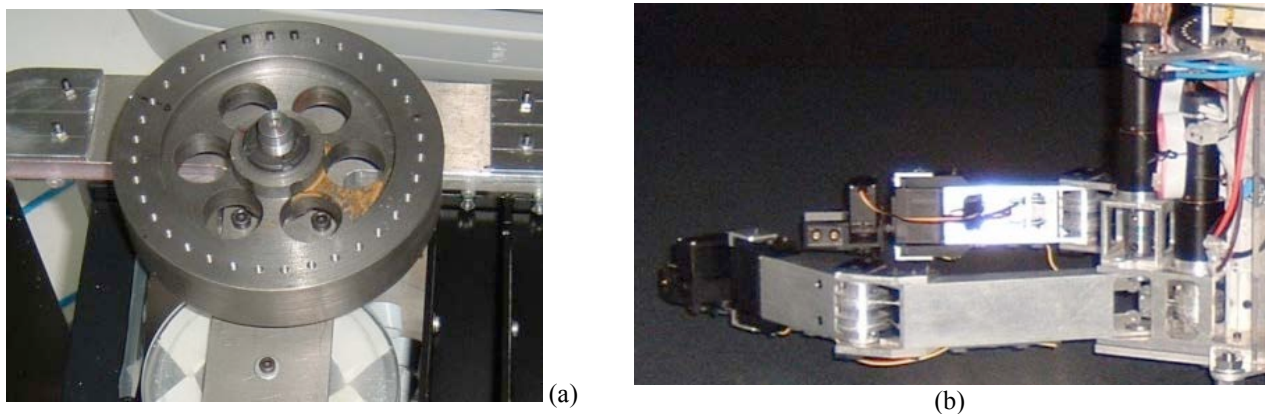


**Figure 8.** (a) The nozzles on the force sensor during tests, (b) The solenoid valves, (c) Experimental results for actuation at 20 Hz and (d) Experimental results for actuation in at 7 Hz.

To make the robot controllable, six nozzles have been installed on the robot, in three counter-facing pairs, 120° apart. More nozzles would increase fuel consumption. However, a provision for two additional valves has been made for redundancy or future needs, while the position of all thrusters can be easily altered if necessary.

(b) *Reaction Wheel*: To reduce gas consumption, a reaction wheel providing torque around a vertical axis was designed and installed, see Fig. 9(a). The wheel was chosen due to the simple control algorithm, attitude fine-tuning capability, and best fit to the two-dimensional experiment. It is actuated by a torque-controlled DC servomotor.

(c) *Robotic Arms*: The robot has two two-dof manipulators, see Fig. 9(b), actuated by DC motors and commanded by the PC104. They have integrated gears for high torque output and are installed all on the main chassis. A sophisticated design employing inextensible micro wires and pulleys enables the independent motion of each link. Each manipulator is equipped with a DC servomotor-actuated and force sensing gripper for capturing of other objects. These grippers can be removed easily and so as to attach an alternative type of gripper or tool.



**Figure 9.** (a) The reaction wheel during early construction phases and (b) The system manipulators.

## **PC104 and RTOS**

An on-board PC104 computer, chosen for its high efficiency, robustness and low power requirements, is used for computational autonomy. The PC104 includes a CPU module (Intel Atom N450, 1.66GHz), two incremental encoder card modules, an analog output module, a wireless LAN module and a SSD module. Each encoder card module is capable of handling up to three incremental 16-bit encoder inputs, has three 16-bit timers and 24 digital I/O lines. The analog output module has eight 12-bit analog output channels and 24 digital I/O lines. The CPU module includes a real time clock and all primary I/O functions (USB, parallel port, serial port, etc.). The PC104 is responsible for motor and thruster control, and reads sensors such as the optical sensors (via the USB port), and the overhead camera (via wireless Ethernet). The PC104 is running under Linux and the programming language of choice is C.

## **4. DEVELOPMENT TROUBLESHOOTING**

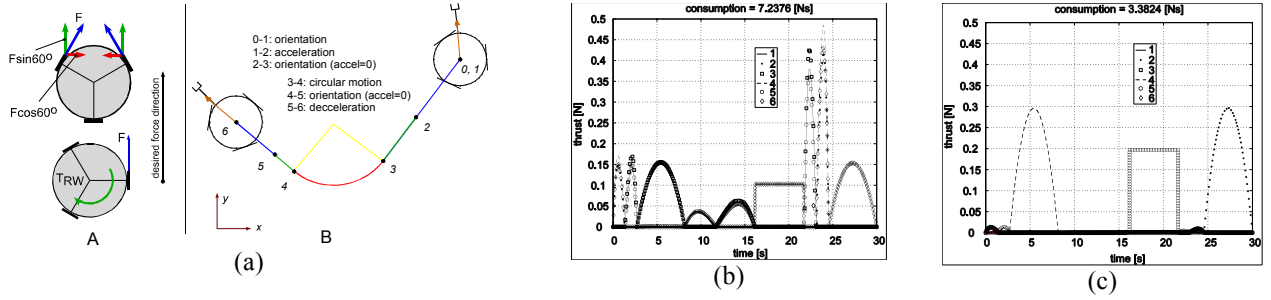
During the development of the robot, various challenges have been arisen. This is justified by the complexity of the system, the interaction of the various components on the confined volume of the robot and the budget constraints which led the team to elaborate several techniques to tackle the problems while on the same time to keep the quality and fidelity of the simulator in the highest possible level. To this end some of the most interesting challenges are presented here:

- (a) Image processing requires the capability to clearly distinguish at least 3 LEDs on top the robot. However in some areas (e.g. at the middle of each side of the granite table) the camera fails to detect LEDs which are installed in the extremes of the top of the robot base. However by adding more LEDs this problem is solved, in the expense of creating a more complex algorithm which is required to detect the exact location of each LED. Currently there are 5 LEDs installed and after experimentation it will be assessed whether this number is acceptable or more LEDs are needed.
- (b) Due to the nature of the image processing algorithm used, several areas of the robot can create «ghost» shines. The camera system could not recognize whether such a shine was a LED or just an indirect illumination. Even by changing the threshold of the halftoning some of these effects could not be eliminated. For this reason there was a redesign of the top of the robot. A black mat PVC board has been selected to cover the robot base. LEDs have been installed on this board. Any remaining shines have been carefully identified and covered.
- (c) Cabling was perhaps one of the more difficult challenges. The confined volume and the complex interconnections between the various subsystem blocks posed several difficulties during the development of the robot. Unfortunately due to lack of experience at the start of the design, there was no particular attention in the requirements for cabling, e.g. specialized routings on the chassis. Due to this fact these routes have been developed later, and several times the installation of cables and CO<sub>2</sub> was a non-trivial task. It was a common event during experiments in integration, where subsystems stopped to function properly, due to small translations of the cables. Therefore it is necessary any new experimental robots to be developed with cabling routes designed from the beginning. It must be also noted that also the mass of the cables (estimated about 1.5-2kg) is an important factor which has not been estimated in the initial stages of the design and development. Particularly the cables where the arms' links rotate pose non-trivial requirements, in order their stiffness not to effect the movement
- (d) During integration it was identified large EMI noise which affected the encoders. The source of the noise has been identified to be the switching DC/DC Converter necessary to provide the 24V power mainly for motors and solenoid valves. Obviously the built-in filter does not cancel out the disturbance. Unfortunately the design requires this converter to be located right in the middle of the motors, therefore a solution must be found. Interestingly enough, during experiments the converted was located outside the robot, therefore minimizing the effects of the noise, and therefore cloaking the severity of the problem which has been discovered only during the integration. Currently several solutions are being sought including among other the development of a simple filter, the purchase of a commercial filter, the use of linear regulators (which on the other hand reduce severely the efficiency), the purchase of another set of DC/DC Converter plus filter or the complete elimination of the converter.

## **5. OPTIMIZATION OF CONSUMPTION: WHEEL VS. THRUSTERS**

Using a detailed model and identified hardware parameters, a number of simulation runs were used to study the minimum fuel consumption problem, and to find desired paths to a final configuration. Even in the optimal case, it is inevitable that the thrust exerted by different nozzles may counterattack each other. It was found that minimum fuel consumption is obtained if full nozzle thrust is used for robot acceleration or deceleration only.

Two generic cases were examined under a typical obstacle avoidance trajectory, see Fig. 10(a). Constraints have been taken into account, such as thruster force limits and torque-speed characteristics. In Case 1, two different thrusters are necessary for the motion. The thruster force components that are perpendicular to the motion do not contribute to robot base acceleration, see Fig. 10(a). In Case 2, a single thruster is fired to accelerate or decelerate the base, while the wheel is used to change the base orientation (to make the thrust parallel to the trajectory) or to cancel undesired torques. The reaction wheel is designed such that the wheel at its nominal rpm can supply the torque required to re-orient the robot in reasonable time. The wheel may saturate when the required torque cannot be produced due to its high rotational speed. In such a case, the propulsion management software checks the wheel status, and if needed, turns the thrusters on. Simulation results were used to study the effects of reaction wheel use on fuel consumption, see Fig. 10(b,c). It was found that Case 2 is fuel-optimal. However, the drawback of this minimum fuel strategy is that the maximum thrust available is that of the thrust of a single thruster, while in Case 1, the thrust is larger (here by 1.4 times), on the expense of higher fuel consumption. Therefore, if economy is important, the wheel must be used, while if performance is important, firing more than one thrusters is preferable. Based on these results, we conclude that under certain assumptions, reaction wheel use is desirable, increasing the life of the robot.



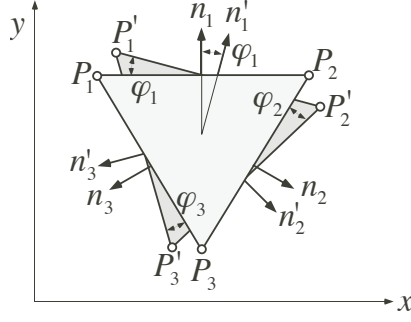
**Figure 10.** (a) Alternative force generation strategies and reference trajectory for comparisons. (b) Thruster forces without a reaction wheel and (c) Thruster forces using a reaction wheel.

## 6. DETERMINATION OF THE ROBOT POSE FROM IMPRECISE POINT POSITION MEASUREMENTS

One of the largest problems for these simulators is the exact localization and estimation of the attitude. As it was explained above two different sensors will be used for this purpose: (a) Relative estimation sensors: the optical sensors, which give fast and accurate estimation but which accumulate drift and (b) Absolute estimation sensor: the camera, which give a slower estimation but with larger accuracy and apparently no drift. Two are the main challenges with this approach. First to determine with sufficient precision the position and attitude of the robot using the data collected and second to combine the information of the various sensors. The latter is a sensor fusion issue, which is very interesting but it is considered as a future work. It is enough to mention that currently the logic is to reset in predefined intervals the estimated position and orientation using the optical sensors' data, by using the estimated position and orientation using the camera. However the former issue, this of the determination of the robot pose from imprecise point position measurement is the scope of this section.

Obviously as any sensor, both optical sensors and cameras have noise in their data. Calibration using various techniques such as estimation of image distortion and calibration using another measurement unit with known (and better) precision, such as Phasespace, will partly mitigate the noise problem. To this end various algorithms have been presented, consider [19] for a further analysis on this. In NTUA a new algorithm has been presented, see [19], which can be used for determining rigid body position and orientation from imprecise points coordinate data. The proposed novel method addresses the kinematic problem of locating a body without defining a body-fixed coordinate system and it has been shown that this method is computationally efficient and yields better orientation estimates than other known methods. Note that the method has been developed for use in 3D rigid body systems, however here only the 2D variant is mentioned, please consult the reference for more into this.

The proposed method for 2D problems (NM-2D) uses a number of coplanar points that define a polygon. Then, the orientation can be found by computing the angle between the normal vector on a polygon's side before and after the rotation, or, equivalently, the angle between the initial and the rotated side. Better than this, the rotation angle can be obtained as the average of all the calculated angles between the original polygon side and the rotated one. For example, if three points are tracked forming a triangle, see Fig. 11, and  $P_i$  and  $P'_i$  ( $i=1,2,3$ ) are points at two consecutive time instants, the rotation angle is given by (2), as follows,



**Figure 11.** Two consecutive time instants for the case of three points position measurements during a rigid body motion.

$$\varphi = \frac{\varphi_1 + \varphi_2 + \varphi_3}{3} \quad (2)$$

where

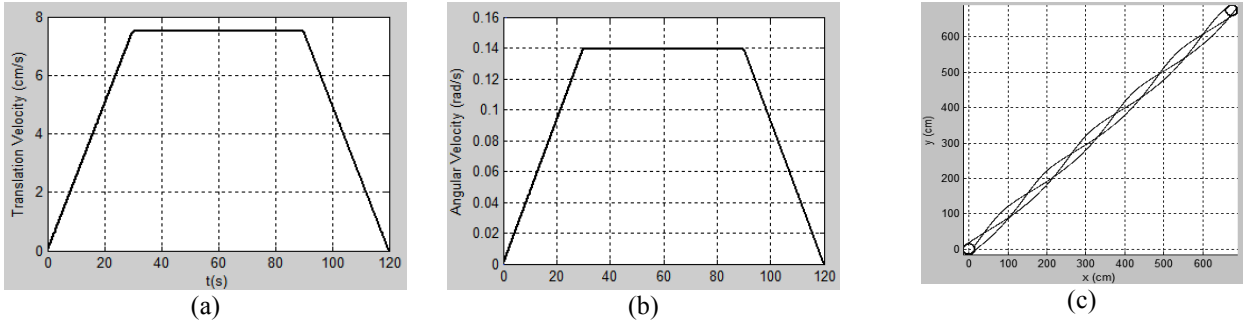
$$\varphi_1 = \cos^{-1} \frac{(\tilde{\mathbf{p}}'_2 - \tilde{\mathbf{p}}'_1) \cdot (\tilde{\mathbf{p}}_2 - \tilde{\mathbf{p}}_1)}{\|\tilde{\mathbf{p}}'_2 - \tilde{\mathbf{p}}'_1\| \|\tilde{\mathbf{p}}_2 - \tilde{\mathbf{p}}_1\|}, \quad \varphi_2 = \cos^{-1} \frac{(\tilde{\mathbf{p}}'_3 - \tilde{\mathbf{p}}'_1) \cdot (\tilde{\mathbf{p}}_3 - \tilde{\mathbf{p}}_1)}{\|\tilde{\mathbf{p}}'_3 - \tilde{\mathbf{p}}'_1\| \|\tilde{\mathbf{p}}_3 - \tilde{\mathbf{p}}_1\|} \quad \text{and} \quad \varphi_3 = \cos^{-1} \frac{(\tilde{\mathbf{p}}'_3 - \tilde{\mathbf{p}}'_2) \cdot (\tilde{\mathbf{p}}_3 - \tilde{\mathbf{p}}_2)}{\|\tilde{\mathbf{p}}'_3 - \tilde{\mathbf{p}}'_2\| \|\tilde{\mathbf{p}}_3 - \tilde{\mathbf{p}}_2\|} \quad (3)$$

If three points are used, the position of point  $C$  is given by,

$$\mathbf{c} = \frac{(\tilde{\mathbf{p}}_1 + \tilde{\mathbf{p}}_2 + \tilde{\mathbf{p}}_3)}{3} = \left( \frac{\tilde{x}_1 + \tilde{x}_2 + \tilde{x}_3}{3}, \frac{\tilde{y}_1 + \tilde{y}_2 + \tilde{y}_3}{3}, 0 \right) \quad (4)$$

while the angle of rotation by (2).

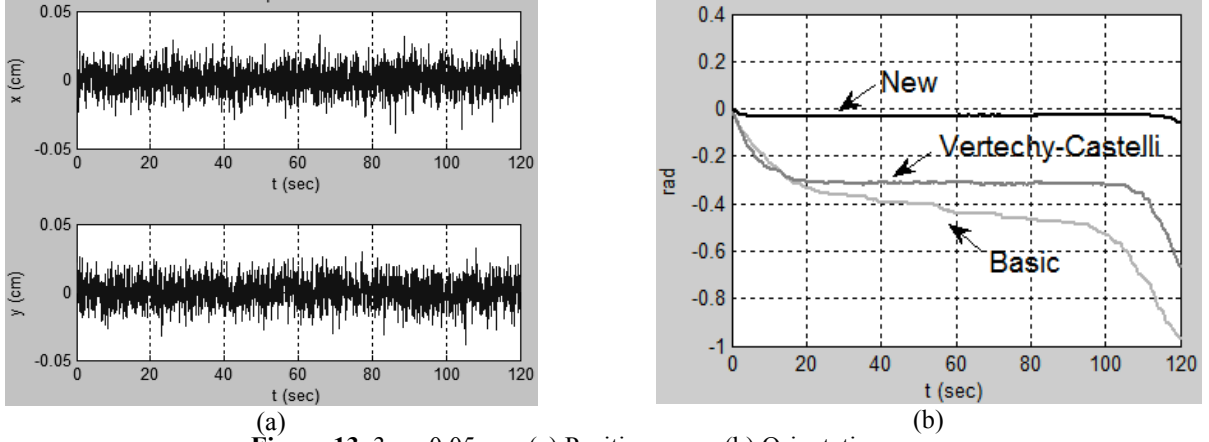
Extensive simulations took place in order to reach reliable conclusions about the behavior of each method in different environments. The methods were implemented in code executable by Matlab. It is assumed that a rigid body performs a complex motion, translating and rotating in 2D for 120 s. The translation and angular velocity trajectories are depicted in Fig. 12(a) and (b), respectively. Fig. 12(c) depicts the rigid body in its initial and final position, as well as the successive positions of points  $P_1$ ,  $P_2$  and  $P_3$ .



**Figure 12.** Body velocity trajectories of (a) translation, and (b) rotation. (c) Initial and final position of the rigid body. Also shown are the successive positions of the three tracked points.

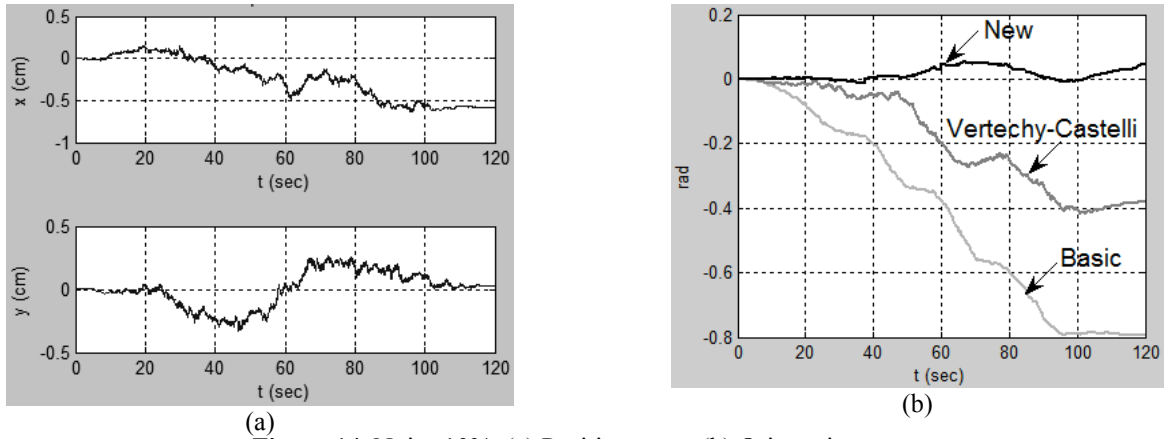
The input data for all methods is the Cartesian coordinates of three non-collinear points of the body. Here, three points were used, placed around a circle of diameter 0.30 m. The exact values of the coordinates of three points were corrupted by adding white noise to simulate the actual measurements, as reflected by a relative or absolute position sensor. The output of all methods is the rotation angle of the body and the position of a particular body point, namely of point  $C$ . The error is considered to be the absolute difference between the actual and the estimated value. It is noted that for the Basic (BM) and Verthey-Castelly (VCM) methods more can be found in [19] and its references.

*Absolute Position Sensor (e.g. Camera):* Fig. 13 shows the position and orientation errors for the case of white noise with mean zero and covariance  $\sigma = 0.01667$  cm. The results of VCM are better than that of BM, while the performance of the NM is by far the best one, evidenced by Fig. 12(b).



**Figure 13.**  $3\sigma = 0.05$  cm. (a) Position error, (b) Orientation error.

*Relative Position Sensor (e.g. Optical Sensor):* Fig. 14 shows the position and orientation errors for the case of white noise with mean zero and covariance equal to 10% times the actual value. With that noise level, BM is almost inapplicable, while results of the VCM method are better. However, the NM performs better than both, as shown in Fig. 14(b).



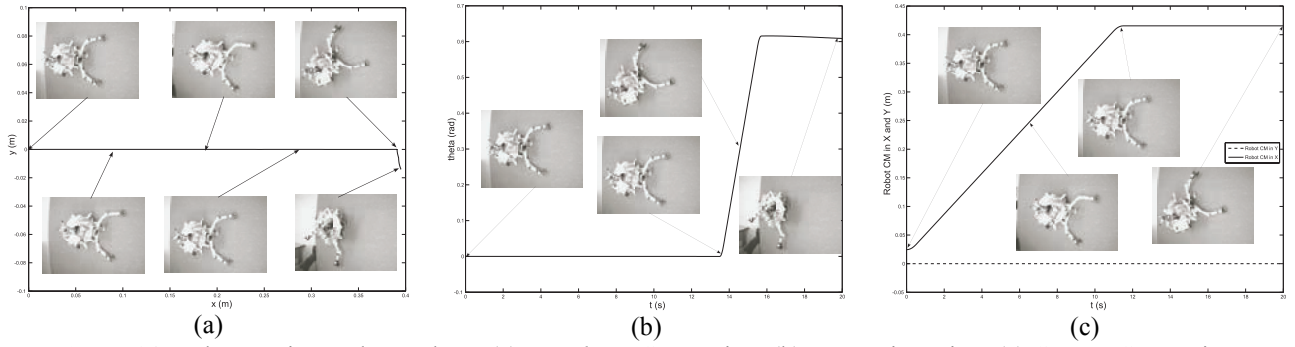
**Figure 14.** Noise 10%. (a) Position error, (b) Orientation error.

In general the NM, both with the absolute and the relative position sensor and for low and high noise level, yields reliable results. This is probably due to the fact that the arithmetic operations in the NM are less, and only absolute differences between noisy measurements are used.

## 7. RESULTS

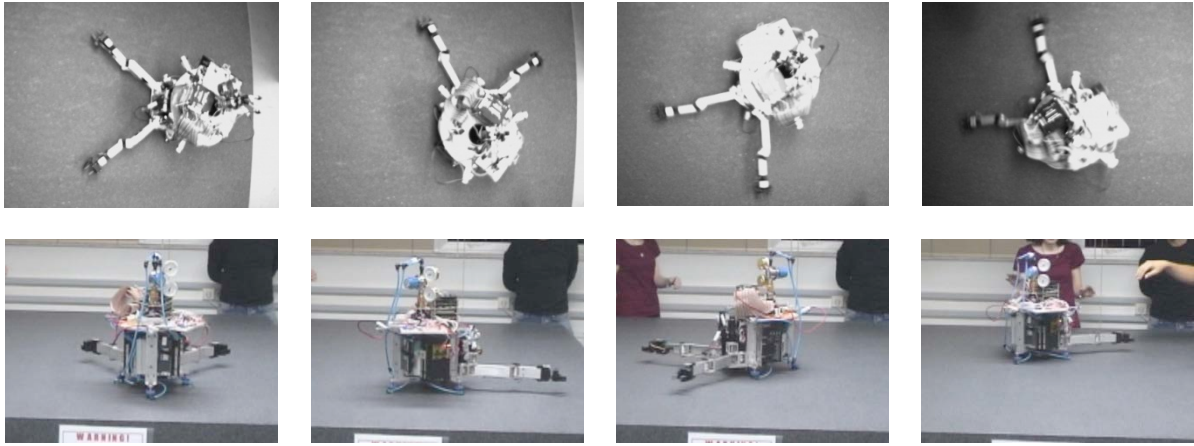
To verify the software simulator and the hardware emulator, a series of experiments were conducted. A motion was simulated in the software simulator, and was also tried in the hardware emulator. The resulting responses were compared to verify the validity of both approaches.

The simulated motion consists of an accelerated forward x-axis motion for 0.5 s. Thrusters are then turned off and the robot moves with constant speed for another 10.5 s. The robot decelerates to standstill by application of opposite thruster force for another 0.5 s. Following a delay of 2 s, the robot rotates using three thrusters that provide a pure torque for 0.2s. The robot is left to rotate around its center of mass (CM) for 1.8 s, when the opposite thrusters are turned on for another 0.2 s, bringing the robot to standstill again. During rotations, the base of the robot not only rotates, but also translates, since the robot revolves around the *system* CM, and not the *base* CM, see Fig. 14.



**Figure 14.** Robot motion and snapshots. (a) X-Y *base* CM motion. (b) *Base* orientation. (c) *System* CM motion.

Although at the time of the first preliminary experiments the hardware was not fully developed yet, an effort was made to capture in hardware the motion described above. The camera feeds image sequences of the experiment. Characteristic snapshots are superimposed with the corresponding simulation responses, see Fig. 14. It can be noted that the emulation is correct: After turning off the thrusters, the system moves in almost pure translation. Small differences between the CAD model, which was used for the calculation of mass and inertia, and the actual hardware affected the motion, which was not completely straight. Additionally, the system slows down before the reversed motion, due to some air-bearings wear, so the simulated standstill in this case is not fully accomplished. However, the rotational motion was emulated very well, with the system staying still following the actuation of the last thruster set.



**Figure 15.** Uncontrollable spinning due to a malfunctioning nozzle. Top snapshots are from the overhead camera.

In another test, a temporary technical problem with the electronic subsystem created a continuous actuation from a thruster, forcing the system in an uncontrollable spinning. This situation however was of great interest, because the system motion resembled the real motion in space, when a spacecraft has a malfunctioning thruster. Fig. 15 shows snapshots of this movement from two different cameras.

Currently, the electronic subsystem is fully developed in the form of PCBs, minimizing the problems arising from the physical construction. The optical sensors provide accurate position versus time, rendering the evaluation of the robot motion easier and more accurate. Finally the robot model will be fine-tuned to minimize differences between the CAD and the final hardware, and the CM location and inertia will be measured accurately.

## 8. FUTURE WORK

Currently the robot is in the final stages of the development. There are two hardware considerations to be dealt with, and minor debugging issues; however the system is operational and ready to run. The two remaining issues are some noise problems caused by a DC/DC Converter and the update of robot CAD model to minimize differences between the CAD and the actual hardware, so that the CM location and inertia will be estimated accurately. The already developed software can move the robot using open loop control. In order to use closed loop control, a procedure is being developed in order for everyone who would like to examine some theoretical approach (e.g. a control theory, an

optimization procedure, a localization algorithm), to be able to do so easily by transferring the theoretical work in specific C files. C libraries that are currently under development and/or debugging will complement the software, for tasks such as data acquisition from the camera and the optical sensors, translation of the desired motion to actuator commands etc. A specific library that must be pointed out is the one merging the optical sensor data, with those of the camera. Since the optical sensor data are acquired quickly enough for control purposes but collect odometric error, the (slower acquired but accurate) camera data are used to periodically correct this error. The use of some MATLAB capabilities is also under consideration and more specifically the use of the xPC target. This will be examined in the near future.

In order to extend the experimental capabilities of the whole simulator, small debris-like systems (passive systems) and new robots (active systems) are being considered. A preliminary design team has been formed to explore the better approach for these systems. Apparently the experience gained and the challenges which have been presented will pave the way for the selection of the optimum components, in order to design and develop a new robot in much less time. Obviously subsystems which are candidates for replacement with more miniaturized and more efficient components are the electronics, the processing system, servoamplifiers and motors. The greatest difficulties arise from the CO<sub>2</sub> subsystem, specifically the storage and distribution subsystem. The new robotic systems will allow the experimentation on various aspects of orbital servicing, such as docking, cooperative manipulation and assembly of large structures and also aspects of swarm robotics. Additionally other interesting experiments would include the systematic procedure of space debris problem mitigation by testing procedures for capturing uncontrollable objects, and the test of reactionless mechanism [20].

Complementary to the abovementioned activities, the creation of a GUI platform in LabView is under development, which would allow a more direct control over the floating systems on the granite table. The user will use an easy interactive method to change specific files (e.g. a control theory implementation), parameters (e.g. gains of a PD controller), have an overall view of all robots and their status (e.g. telemetry of the sensors), have real time analysis of the image processing etc. This way a dedicated “control-room” of the space simulator will be created, and therefore other aspects, such as ergonomic issues or teleoperation (with or without delay) could be tested.

## 9. CONCLUSIONS

The NTUA software simulator and the hardware emulator for space robots studies were presented. The software simulator was designed to be fully parametric, enabling the user to simulate generic designs of real space robot concepts. The program allows the derivation of the system dynamics, and the testing under various control schemes in 3D motion. The motion is animated in 3D for better response evaluation. The planar hardware emulator enables experimentation in an emulated zero-g, 2D environment. The robot is mounted on air bearings, is propelled by PWM thruster forces, and includes two 2-dof manipulators with actuated grippers. Preliminary results were presented, where the simulated response was compared with the hardware experiment. These showed the potential that this system has for studying space robots in orbit.

## 10. ACKNOWLEDGEMENT

Part of this research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitos II. Investing in knowledge society through the European Social Fund.

## 11. REFERENCES

- [1] Yoshida, K., “Space Robot Dynamics and Control: To Orbit, From Orbit, and Future”, *Robotics Research, the Ninth International Symposium*, ed. by J. Hollerbach and D. Koditschek, Springer, pp. 449-456, 1999.
- [2] Christiansen, S. and Nilson, T., “Docking System for Autonomous, Un-manned Docking Operations”, *Aerospace Conference 2008 IEEE*, 1-8 March 2008.
- [3] Howard, R.T., Heaton, A.F., Pinson, R.M., and Carrington, C.K., “Orbital Express Advanced Video Guidance Sensor”, *Aerospace Conference 2008 IEEE*, 1-8 March 2008.
- [4] Ogilvie, A., Justin Allport, Michael Hannah, John Lymer, “Autonomous Satellite Servicing Using the Orbital Express Demonstration Manipulator System,” *Proc. of the 9<sup>th</sup> International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS '08)*, Los Angeles, California, February 25-29, 2008.
- [5] [http://www.nasa.gov/mission\\_pages/dart/main/](http://www.nasa.gov/mission_pages/dart/main/)
- [6] Tatsch, A., Fitz-Coy N., Gladun S., “On-orbit Servicing: A Brief Survey”, *IEEE Int. Workshop on Safety, Security & Rescue Robotics, NIST*, Gaithersburg, MD, August 22 - 24, 2006.

- [7] Rouleau, G., Rekleitis I., L'Archeveque R., Martin E., Parsa K., Dupuis E.: "Autonomous Capture of a Tumbling Satellite", *IEEE International Conference in Robotics & Automation*, Orlando, FL, pp. 3855 – 3860, May 2006.
- [8] West, H., Papadopoulos, E., Dubowsky, S., and Cheah H., "A Method for Estimating the Mass Properties of a Manipulator by Measuring the Reaction Moments at its Base," *Proc. of the IEEE Int. Conference on Robotics and Automation*, Scottsdale, AZ, pp. 1510-1701, May 1989.
- [9] Schwartz, J., Peck M., Hall C., "Historical Review of Air-Bearing Spacecraft Simulators", *Journal of Guidance, Control, and Dynamics*, vol. 26, no. 4, pp. 513-522, 2003.
- [10] Wertz, J, Chen. A, et al., "SPHERES: A Testbed for Long Duration Satellite Formation Flying in Micro-Gravity Conditions", *Proc. of the AAS/ AIAA Space Flight Mechanics Meeting*, Clearwater, FL, January 23-26, 2000.
- [11] Mohan, S., Sakamoto H., and Miller D. W., "Formation control and reconfiguration through synthetic imaging formation flying testbed (SIFFT)", *SPIE Optics & Photonics*, San Diego, CA, 26-30 Aug 2007.
- [12] <http://sun-valley.stanford.edu/>
- [13] Adams, J., Robertson A., Zimmerman K., How J., "Technologies for Spacecraft Formation Flying", *Proc. of the Institute of Navigation GPS-96 Conference*, pp. 1321-1330, Kansas City MO, September 1996.
- [14] Bettanini, C., et. al., "Improving the Free-floater Space Robot Simulator for Intervention Missions", *Proc. of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2003)*, NARA, Japan, May 19-23, 2003.
- [15] [http://www.sesnet.soton.ac.uk/people/smv/avs\\_lab/index.htm](http://www.sesnet.soton.ac.uk/people/smv/avs_lab/index.htm)
- [16] Nakamura, Y., Sasaki F. and Nakasuka, S., "Guidance and Control of Tethered Retriever with Collaborative Tension-Thruster Control for Future On-Orbit Service Missions", *Proc. of the 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2005)*, Munich, Germany, Sept. 5 - 8, 2005.
- [17] Zhang, Z. Y., "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330-1334, Nov. 2000.
- [18] Russ, J. C., *The Image Processing Handbook*, CRC Press, 2006.
- [19] Tegopoulou, A. and Papadopoulos, E., "Determination of Rigid-Body Pose from Imprecise Point Position Measurements," *Proc. of the 2011 International Conference on Intelligent Robots and Systems (IROS '11)*, September 25-30, 2011, San Fransisco, CA, USA.
- [20] Andreou S. and Papadopoulos, E., "Design of a reactionless pointing mechanism for satellite antennas," *11th ESA Workshop on Advanced Space Technologies for Robotics and Automation (ASTRA 2011)*, 12 – 14 April 2011, ESA/ESTEC, Noordwijk, The Netherlands.