Rescue Robot League Competition
Bremen Germany
14-20 June 2006

RoboCupRescue 2006 - Robot League Team Ariana (Iran)

Mohsen Rahnavard[1], Ahmad Chitsazan[2], Amir Hossein Soltanzadeh[3],

Pedram Johari[4], Ehsan Abbasi[5], Mahdi Emami[5], Mahdi Ramezani[5]

[1] Shahed Research Inc.
86 Shahed Research Inc. Bldg. 1F
Vesal, Tehran, Iran
mohsenrahnavard@dsprt.com
http://robot.dsprt.com

[2] DSP Research Co.
747 Bahar Bldg. 10F
South Bahar, Tehran, Iran
chitsazan@dsprt.com
http://dsprt.com

[3] Xarrin Advanced Technologies
1 Sabz Bldg. 7F
South Shiraz, Tehran, Iran
amirhst@xarrin.com
http://xarrin.com

[4] I. Azad University Tehran Central Branch Vali-Asr Complex
Damavand, Tehran, Iran
pedrram@yahoo.com

[5] I. Azad University Tehran Central Branch Vali-Asr Complex
Damavand, Tehran, Iran

Abstract. In this paper we have introduced our newest product (Arian III) outlines. It's a track based robot with two separately controlled flippers. Actually it's a *semi autonomous* system with reliable control methods for navigation in mission operation. Although till now our group products have been manufactured as industrial products for real environmental conditions, it's to our pleasure that we have found a reliable arena for demonstration of our capabilities.

Introduction

Ariana robotic group was founded to industrialize rescue robots and extend their capabilities for real natural urban disasters at 2002. Arian III is the youngest member of Arian's family which all members are track based robots. It is a robust track robot for operating in variable undesirable conditions such as mechanical and thermal shocks, humidity, electrical noises, X radiations...
To control the stability of robot in two directions (along the length and width of robot), Arian III benefits two separately controlled flippers.
It takes advantage of "the state of the arts" control unit for simplifying the operator's tasks in real disaster conditions.
Arian III has three control layers *(Manual, Semi autonomous and full Autonomous)* to operate in different environmental situations.

Map generation and localization is done *full automatically* without any interfering of the orator. Also victim identification is done automatically under the supervision of the operator.

1. Team Members and Their Contributions

Our team consist of some specialist in various fields :

• Mohsen Rahnavard (Ph.D. of Solid State)	Team Leader & Embedded Sys.
• Mazyar Sadeghi (M.S. of Mech. Eng.)	Mechanical design
• Amir Hossein Soltanzadeh (B.S. student)	Mechanical design
• Reza Jalili Saffar (M.S.student)	Mechanical Manufacture
• Ahmad Chitsazan (B.S. student)	Mechanical Manufacture
• Farzin Mozaffari (B.S. student)	Mechanical Manufacture
• Ahmad Byagowi (M.S. student)	Network & Communication
• Mahdi Emami (B.S. student)	Navigation & Map generation
• Ehsan Abbasi (B.S. student)	Sensors & Power supply
• Pedram Johari (B.S. student)	Control Developer
• Mahdi Ramezani (B.S. of Computer Eng.)	Vision & Driver
• Asghar Mirzaei (B.S. of Computer Eng.)	GUI & Software Developer
• Hamid Aeinehsaz (M.A. of Interior Arch.)	Advisor
• Hossein Mahbadi (Ph.D. of Mech. Eng.)	Advisor
• Zaki Byagowi (Ph.D. of Electro technique)	Advisor

2. Operator Station Set-up and Break-Down (10 minutes)

After some sessions with a trained rescue team at Helal-e-Ahmar Org. (Iran Red Cross) we were acknowledged that a rescue teams consist of three specific groups:

- Advance team (just emergencies)

- Support team (ambulances, helicopters, …)
- Back up team (tracks, debarkation hospital, …)

Rescue robots mostly are used in advanced team and because of distance should be traveled by the rescue men carrying robot on foot, rescue robots must be portable. We've conceded this particular point in all design procedures and have used two kinds of packing, first for traveling to disaster zone and second for carrying the package to the mission location.

2.1 Travel Packing

Arian III travel package consist of three suitcases (see Fig.1):

- Main unit (28 Kg)
  - Robot main body (exc. VL parts)
- Control Unit (3.5 Kg)
  - Embedded main board (PIII)
  - Touch panel display
  - Heads up Goggle
  - Industrial joystick
  - Keyboard
  - Stereo speaker
  - Microphone
- Equipment unit (12 Kg)
  - VL parts
  - Battery packs
  - Battery charger
  - Solar cells
  - Mechanical Adjustable Tools

Fig.1 Traveling Package of Arian III (to behind: Control unit, Equipment unit and Main unit)

2.2 Carrying Packing

- Arian III (backpack 29 Kg)
- Control Unit (suitcase 3.5 Kg)

Traveling pack can be transported by two people to the disaster zone. At disaster zone, after assembling VL and MAD parts on the main body, operator will be able to put the robot in the back pack and pick up the light control unit by hand.

2.3 At Robocop competitions

The traveling package of Arian III is so portable that there is no need back pack for transportation.

Operator Station Set-up (10 minutes)

Robot can be carried by one person and another needed for transporting the control unit. As mentioned above, control unit is a squeezed pack containing embedded mother board (Eden/C3 Embedded SBC equivalent PIII 677MHz), LCD touch screen display, keyboard, joy stick, heads up, stereo speaker, microphone; so the operator should just turn the robot and control unit on and supervise the automatic system tests. The automatic system test procedure is consisted of these steps:

- Testing all connection system actual
- Making an allowance for the motors receipt feed back by giving the short commands to the motors
- Examining sensor connection or disconnection with considering their received quantities
- Testing The robot sight by examining the camera signal receives
- The system localization test and its calibration

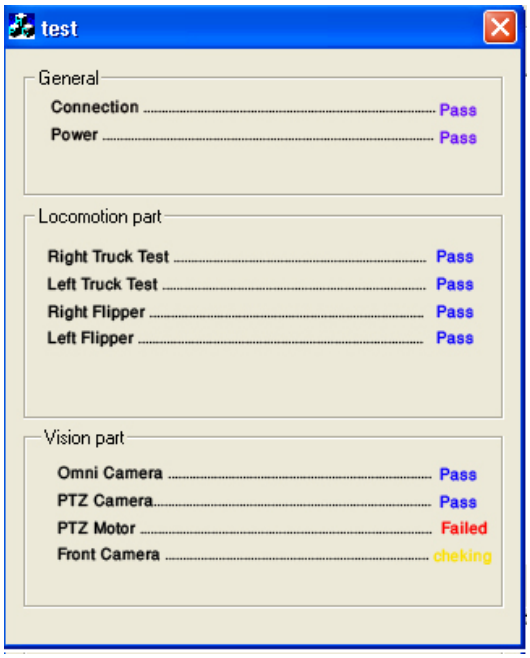


Fig.2 System checking at the start up

2.3.1 Operator Station Break-down (10 minutes)

By ending the match time, the operator has two main tasks: first making the control unit to print generated map and second, ordering to the robot for coming back automatically by using recorded path way in robot memory.

3. Communications

After years of experience in participating in Robocop rescue leagues, today we have decided to do most of our tasks *semi-autonomously*, which will be more described in the 4th section. Since yet there are some commands and tasks that should be held on by the operator, we need a good and reliable communication protocol, and also we know that data have to be exchanged in both directions as the robot has to be controlled and to send back video and sensors data.

- **Data to be transmitted to the robot:**
  - ✓ Motor control
  - ✓ Camera control
  - ✓ Operator's sound

- ✓ Information request

- **Data to be received:**
  - ✓ Sensors data
  - ✓ Images (streaming video)
  - ✓ Microphone sound

So we reached to the idea of using two way communications that are described later. First one is the **Wireless LAN** *(part 3.1.1)* whit a powerful Access-Point and a good antenna to sending all the controlling data, and receiving the whole information about the sensors, robot situation, and streaming videos of our cameras; and the other one is a **Radio Frequency Transceiver** *(part 3.1.2)* of 2.4GHz frequency which will transmit the control information and receive the main camera's streaming video in the critical situation when the wireless LAN fails.

In the sections below we have described the communication system in two separate parts which are **remote** and **internal** communication.

3.1 Wireless (remote) Communication

To exchange (send/receive) data with the robot we have to use a remote (wireless) communication, which will be described as two different methods of send/receive data between the robot and the control unit.

Also we have a remote connection between the joystick and the control unit which is described in section 3.1.3.

3.1.1 Wireless LAN

The wireless LAN which we use is working on the **802.11A (5GHz)** protocol link, and it can be set to a stay on a specific channel. 802.11a uses a different frequency than the most commonly used 802.11b/g and will procure a more robust connection. But also we can switch to the **802.11B/G (2.4GHz)** if necessary. At a higher level, UDP sockets will be used instead of TCP sockets which are less efficient in bad network conditions. Thus the application software will have to take care of all the streaming synchronization.

While traveling up the TCP stack, you might have wondered when you'd get to the point of doing something useful. Pinging and routing are all very fine, but you can't use them to exchange meaningful data with another system. In our rescue robot system what is really needed is a simple protocol that sits on top of IP and allows you to launch data and receive replies on your network. That protocol is user datagram protocol (UDP), so we will use the UDP instead of TCP.

3.1.1.1 Access Point

We use a DWL-7200AP D-Link access point. The picture and the specifications of this part are attached below.

Fig.3 DWL-7200AP Access Point

Table .1 DWL-7200AP AirPremier AG Tri-Mode Dualband 802.11a/b/g (2.4/5GHz) Wireless 108Mbps[1] Access Point with PoE Specifications.

Specifications	
Standards	• IEEE 802.11a • IEEE 802.11b • IEEE 802.11g
Radio and Modulation Type*	For 802.11b: DSSS : • DBPSK @ 1Mbps • DQPSK @ 2Mbps • CCK @ 5.5 and 11Mbps For 802.11a/g: OFDM: • BPSK @ 6 and 9Mbps • QPSK @ 12 and 18Mbps • 16QAM @ 24 and 36Mbps • 64QAM @ 48, 54 and 108 Mbps DSSS: • DBPSK @ 1Mbps • DQPSK @ 2Mbps • CCK @ 5.5 and 11Mbps
Wireless Signal Range*	802.11g (Full Power with 5dBi gain diversity dipole antenna) Indoors:

	• 98ft (30m) @ 54Mbps • 105ft (32m) @ 48Mbps • 121ft (37m) @ 36Mbps • 148ft (45m) @ 24Mbps • 203ft (62m) @ 18Mbps • 223ft (68m) @ 12Mbps • 253ft (77m) @ 9Mbps • 302ft (92m) @ 6Mbps Outdoors: • 328ft (100m) @ 54Mbps • 968ft (295m) @ 11Mbps • 1378ft (420m) @ 6Mbps
Transmit Output Power	For 802.11a: • 32mW (15dBm) • 6mW (7dBm) For 802.11b: • 10mW (10dBm) • 6mW (7dBm) For 802.11g: • 32mW (15dBm) • 6mW (7dBm)
Temperature	• Operating: 32ºF to 104ºF (0ºC to 40ºC) • Storing: -4ºF to 149ºF (-20ºC to 65ºC)
Humidity	• Operating: 10%~90% (non-condensing) • Storing: 5%~95% (non-condensing)
Dimensions	• L = 6.89 inches (175mm) • W = 4.13 inches (105mm) • H = 0.79 inches (20mm)
Weight	• 0.44 lbs (200g)

Since the antenna which is installed on the Access Point is very weak and can be harmed easily, we use a kind of antenna which is more useful in tumultuous areas.

Fig.4 ANT24-0800

Table .2 ANT24-0800 Antennas Outdoor Omni-Directional Antenna specifications

Electrical Specifications	
Frequency Range	2.4 -2.5GHz
Gain	8 dBi
VSWR	2:1 Max
Polarization	Linear, vertical
HPBW	• horizontal - 360° • vertical- 15°
Downtilt	40°
Power Handling	50W (cw)
Impedance	50 Ohms

3.1.2 Radio Frequency Transceiver (RFT)

When the wireless LAN is failed our controlling unit will announce the operator that there is a problem in send/receive process with the wireless LAN, and he will change the communication mode to the RFT. In this mode we can only send the control commands of driving the robot and the main camera and receive the pictures of the main camera which is installed in the front of the robot. During send/receive with this method the wireless LAN will be checked every 100msecs and will be automatically selected if there isn't any more problems.

Fig.5 VRX-1290LX

This is a newest Audio / Video transceiver for 2.4 GHz band. Receiver is fully synthesized VFO type for a full range of 2,300 MHz- 2,500 MHz with amazing sensitivity -96dBm/ 10 dB. Transmitter section has 8 channels and 250 mW RF power. Transceiver has two separate RF inputs and it works from 12 V / 750 mA. It is easy to use. The range of this device is over 5 km line-of-sight with special High gain antennas. It is excellent for two-way Audio / Video communication point-to-point.

Table .3 RECEIVER SPECIFICATIONS:

BATTERY POWER: 12 V - 16 V	PLL (PHASE LOOKED LOOP) CONTROLLED!
VIDEO OUT 1V PEP	SIZE: **6.5" X 4" X 1"**
CURRENT CONSUMPTION: 300 mA!	SMA ANTENNA CONNECTOR
SENSITIVITY -96 dBm/10 dB SIN	AUDIO OUTPUT 300 mV
FULLY TUNABLE VFO IN 2.3- 2.5 GHz BAND	BUILT-IN RF AMPLIFIER
FM DEMODULATION	FM BANDWIDTH 16 MHz

Table .4 TRANSMITTER SPECIFICATIONS:

BATTERY POWER: 12 V - 16 V	PLL (PHASE LOOKED LOOP) CONTROLLED!
VIDEO INPUT 1V PEP	SIZE: **6.5" X 4" X 1"**
CURRENT CONSUMPTION: 180 mA!	SMA ANTENNA CONNECTOR
RF POWER 250 mW/ 50 ohms	AUDIO INPUT 2 mV
8 SELECTABLE CHANNELS	BUILT-IN FILTER
FM DEMODULATION	BUILT-IN PROTECTION CIRCUIT

3.1.3 Bluetooth telecommunication between the joystick and the main control unit

Since we have two remote control units for different situations and one of them is packed in a rucksack we have to use a Bluetooth communication between the joystick and the main control unit to avoid using of wire and more complexity in the control unit.

In this part we use a Bluetooth module on the joystick which will send data to our laptop in the rucksack and our driver for this Bluetooth module will translate the codes which we have sent with the joystick for our software to do the related task.

3.1.4 Table of the frequencies that apply to our team

Here is the table of the frequencies that apply to our team, and a conclusion to all the three parts above.

Table. 5

Rescue Robot League		
ARIANA(Industrial Rescue Robot) (IRAN)		
MODIFY TABLE TO NOTE ALL FREQENCIES THAT APPLY TO YOUR TEAM		
Frequency	**Channel/Band**	**Power (mW)**
5.0 GHz - 802.11a	any	8
2.4 GHz - 802.11b/g	any	8
2.4 GHz - Bluetooth	spread-spectrum	
2.4 GHz – Audio/Video Transceiver	8channels/ 2300MHz~2500MHz	250

3.2 Internal Communication

Our internal devices in the robot are connected to each other with a local area network (LAN) which will finally be sent to the control unit via the wireless LAN either the RF transceiver.

The whole communication system is briefed in the Block Diagram below:

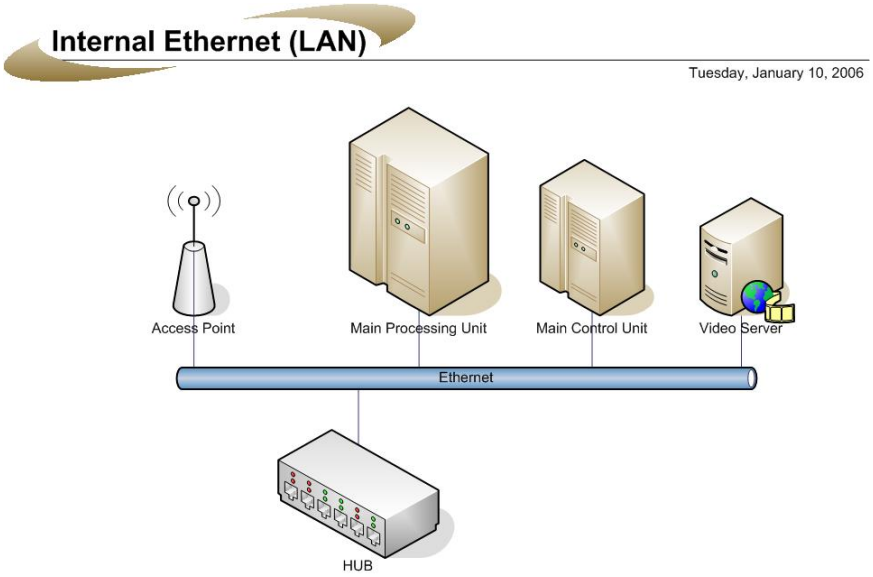


Fig.6 The block diagram of the Internal Ethernet of Arian III

The video server has four analog inputs and it captures and quads the inputs' image and send it through the Ethernet.

Fig.7 Video Server

4. Control Method and Human-Robot Interface

Nowadays one of the most important problems in robotic researches is how to control the robot best way. So to have a good robot, we should have a good control on it. To control Arian III we have provided a powerful software to be efficient in any situation.

As we mentioned in previous parts of this TDP this year we have decided to do most of our tasks *semi-autonomously*, such as controlling the flippers of the robot automatically with the information which we gather from the sensors that are installed on the flippers and sensing the torque (strain gage), and automatically control the angle of the flippers.

4.1 Control Method

Below we have illustrated our total plan briefly in a block diagram.

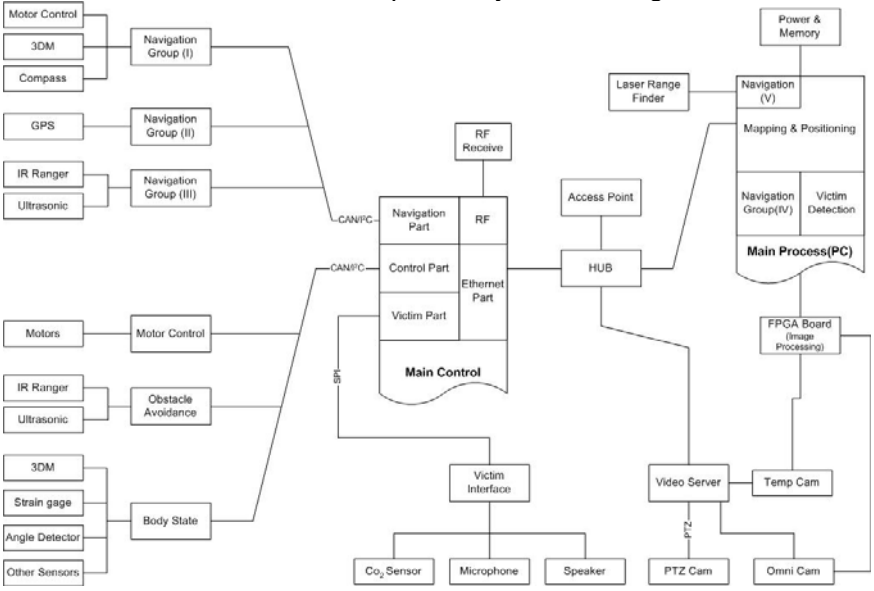


Fig.8 The control diagram of Arian III

4.1.1 Main Control Unit

The main control board has to do four different tasks:

- ✓ Receiving the data from three navigation groups and sending these information on the Ethernet for the main processing unit

- ✓ Receiving the data of victim identification groups and send them through Ethernet for the operator
- ✓ Receiving the control data which is sent through RF transmitter
- ✓ Semi-autonomous controlling of robot according to the received control and sensors data

Fig.9 Main Control Board of Arian III (Embedded LAN Control Board)

4.1.2 Main Processing Unit

The main processing unit contains an industrial embedded main board with a [Eden/C3 Embedded SBC Equivalent PIII 677MHz] CPU, Touch screen LCD, hard disk, and the power supply.

This part has to do these tasks

- ✓ Gathering the information of the five navigation groups and generating the final map
- ✓ Processing the raw data of the fourth and fifth navigation groups
- ✓ Processing the image provided by the thermal camera

Fig.10 Arian III ‘s main processing unit

4.1.3 Motor Controller Board

One of the most important parts of the robot is the motor controller board. We have designed a good motor controller which is derived from the MD03 motor controller board. The PCB of this controller is attached below.

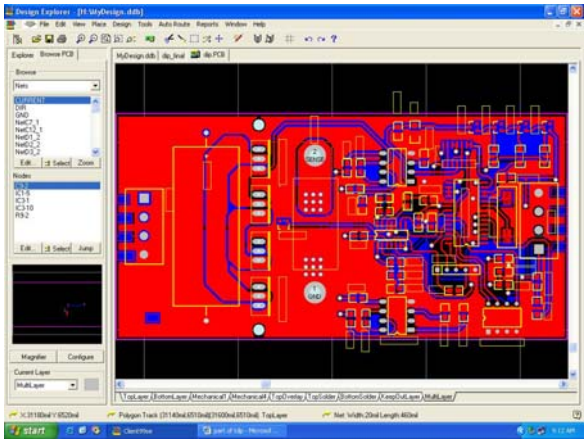


Fig.11 PCB of Arian III ‘s motor controller

Fig.12 Arian III ‘s motor controller

4.2 Human-Robot Interface

One operator will control the robot via a PC interface. All functions will be accessed with a joystick and finger touching (we use touchpad LCD).

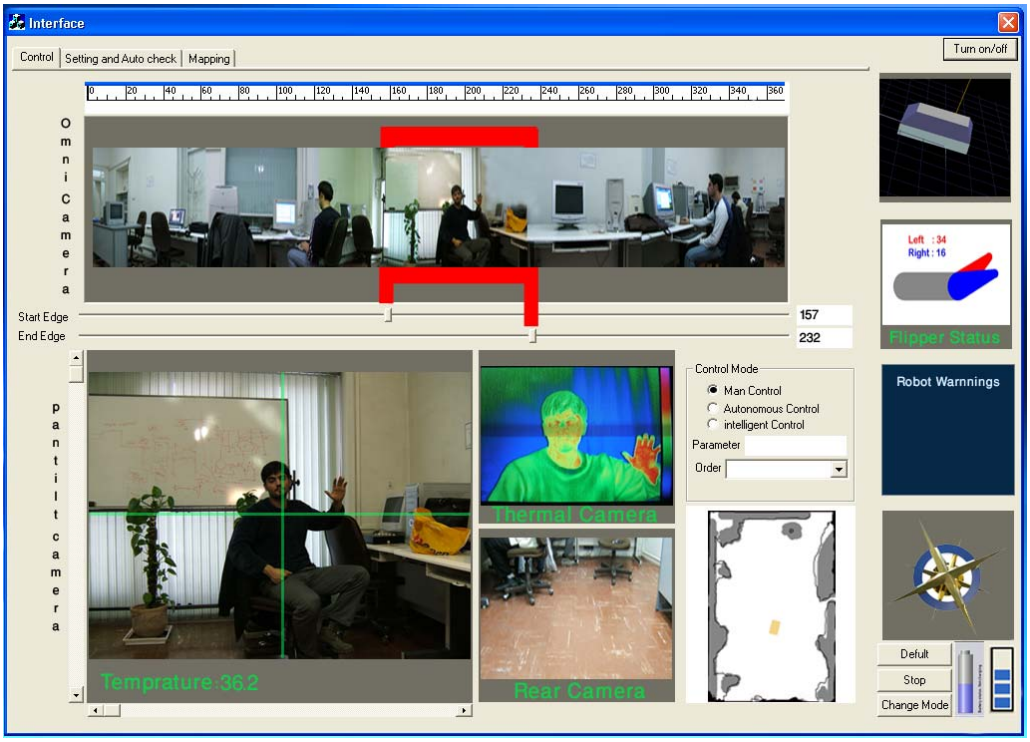


Fig.13 Interface Overview

Control tab contains

- Omni directional camera view
- Pan Tilt camera view
- Thermal Camera view
- front Camera view
- Control mode
- 3D robot spatial orientation view
- Flippers status
- Robot warnings
- Compass
- Power meter
- Connection status

4.2.1 Omni directional camera view

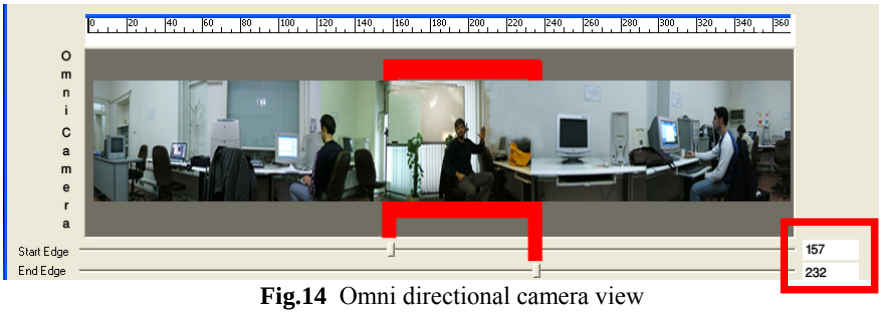


Fig.14 Omni directional camera view

In this part operator view 360 digress of robot's environment and it is possible to select the part of environment which he like to see with pan tilt camera, also the range be shown with 2 "Edit Boxes" near to "start edge" and "end edge".

4.2.2 Pan tilt camera view

Fig.15 Pan tilt camera view

Pan tilt camera view; automatically show the selected area that is chosen in omni directional view. But operator can reject this capability and chose area with scroll bars.
In this part, the temperature of center of image is explained.

4.2.3 Thermal camera view

Fig.16 thermal Camera view

Thermal camera view has 3 parameters need to set; X-Y position and zoom condition, all of these fields are controllable with 3 scroll bars.

4.2.4 Front camera view

Fig.17 Front Camera view

Front camera is a fix camera that shows front of the robot. Therefore there is no parameter to set.

4.2.5 Map view

Map view monitors robot and all part of map that are created.
Additional information about map is available in map tab. Map view in control tab is an assistant tools for operator to have a better imagination of robot's position.
Map view has zoom capability.

Fig.18 map view

4.2.6 Control mode

This robot has 3 control levels:

1- *man control*
2- *autonomous control level*
3- *intelligent control level*

In man control level, operator use interface information contains videos, sensors information, robot warnings and etc. to handle robot with low level commands such as "forward, backward, stop …" but this instructions can be commanded by joystick and keyboard to reduce operator's difficulty.
Autonomous control level, aid operator to order easier than man control level. In this situation operator use high level commands to order robot like "go ahead 4 meters". This case cause operator has more time to attention to victim identification and improves efficiently of operator.
All instructions are available in a combo box and parameters specify whit edit box.

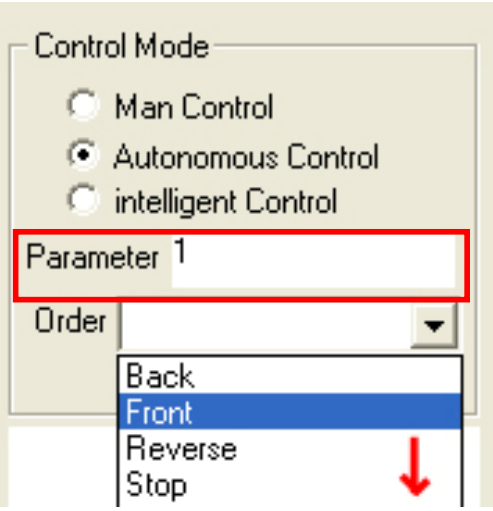


Fig.19 autonomous control level

In intelligent control mode, operator only specific movement radius of the robot and robot check specified area to identify victim and generate a map.
After checking area, robot sends a signal that show end mission. Clearly because of limitation of this technique operator can change this mode rapidly when he wants to man control level or autonomous control level.

4.2.7 3D robot spatial orientation view

This part monitors robot's spatial orientation based on result of 3DM sensor.

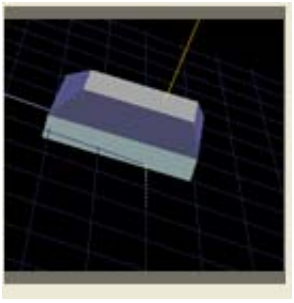


Fig.20 spatial ordination view

4.2.8 Flippers status

Flippers status is a graphical interface that is monitored 2 flippers of robot and angles.

Operator can control each flipper angle independently or synchronal.

Fig.21 Flippers status graphical interface

4.2.9 Robot warning dialog

Robot warning dialog is a text base dialog that monitors all signals is received from robot in critical situation.

4.2.10 compass

Compass figure shows robot direction.

Fig.22 compass figure

4.2.11 Power meter

Power meter monitors battery charging.

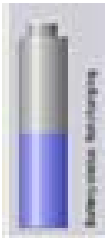


Fig.23 Power meter

4.2.12 Connection status

Connection status monitor connection signal between robot and controller.

Fig.24 Connection status graphical interface

5. Map generation/printing

There are various methods of mapping and localization which have their own special errors. After studying about these methods we have came to conclusion that to have a fair map we need fine localization and vice versa. It means that these two subjects are not separable and each one depends on the other. So we have tried to solve these problems simultaneously.
N the other hands, as mentioned each method has its own errors which depend on some parameter. These parameters are:

- ✓ The environment in which the robot works
- ✓ Sensors which are used in the method
- ✓ Nature of the method

Therefore to have a map and position with least error we have used a combination of different methods, and we gather the information of each method separately and combine the results of them with a special algorithm and then generate the map and derive the position of the robot. This way, different methods with different errors will cover each other's faults and the map and position which is obtained in each step contains the east error. For instance in one method has absolute error and the other has accumulative error, and one system has static error and the other has dynamic error.
It's necessary to be mentioned that all the process of mapping and localization will be done *automatically* and uses *AI* and there is no need for the operator to do anything, and only the final result of each step will be displayed on the LCD for the operator. Also as it will be mentioned later at the victim identification section, finding the victim with the sensors will be done *full-autonomous* and will be added to the map in its position.

5.1. Navigation Groups

To provide the map we have used five different methods in five separate groups. Each group contains a processor and some sensors that will independently trying to create map and cover the position of the robot.
Studying different existing theories, we have tried to decrease the error in each group to the least possible value, and do the operation needed in each group in best possible way, and finally at the specific time periods, each group will send its data to the main processor which is a Pentium III with a coefficient. This coefficient which is called assurance coefficient is the assessment of each group's processor for the accuracy of their own sent data to the main processing unit.
In each group the coefficient will be made with a special method by its processor by use of some parameters. The pent-groups and their members are:

- ✓ First group
    I. GPS

- ✓ Second group
    I. Compass

II. Gyroscope (accelerator-magnetometer-…) 3DM
III. Tachometer
IV. Feedback of motor's current

✓ Third group
I. Ultrasonic
II. IR ranger

✓ Fourth group
I. Laser range finder

✓ Fifth group
I. Omni-directional camera

The main processor will receive each group's data and assurance coefficient in every period and after correcting this coefficient by using a fuzzy algorithm and Kalman filter, will combine the data and coefficients and then making up the map and decide the position of the robot in the period. After formation of map and position in each step, the data will be transmitted to the operator. And also some information for correcting the errors will be sent to the processor of each group. The block diagram of the method of operation for the groups and the main processor in mapping and localization:

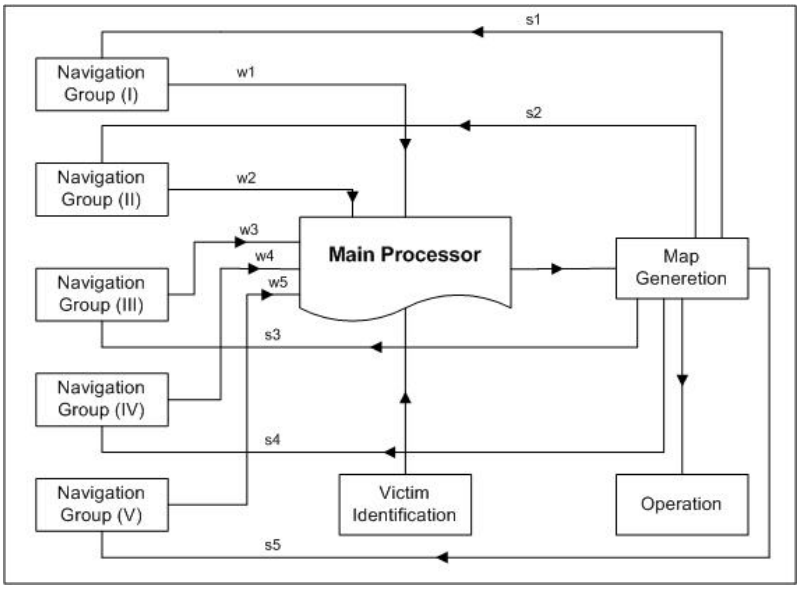


Fig.25 The block diagram of method of mapping and localization where w_i are the assurance coefficients and s_i are the data needed to correct the errors.

5.2. Modifying the assurance coefficient

The main processor needs some information about the environment of the robot to be able to modify the coefficient.
We have divided the specifications of the environment into 4 groups:

- ✓ **First**
  1. Indoor
  2. Outdor
- ✓ **Second**
  1. Structured
  2. Unstructured
- ✓ **Third**
  1. static
  2. dynamic
- ✓ **fourth**
  1. small scale
  2. large scale

The main processing unit is able to recognize the environment specifications with some special algorithms automatically. The origin of this classification for environment is the ability of the robot to be used and work in various environments. Since our robot is designed to be used in industrial and military applications, it should be able to work in various kinds of environments. Besides, the accuracy and efficiency of each method and its sensors in disparate environments is different. For instance GPS will cause more error in indoor usage than outdoor, or the range finders will work more reliable in structured areas than unstructured ones. Therefore the main processor should correct and modify the sent coefficients of the methods, referring to the environment in which the robot works, to combine the methods and generate a fair map.
The main processor uses a specific method in each classification, to recognize each kind of environment. Naturally the process of recognizing the environment will be completed after several time periods. One way to recognize the areas is to compare the assurance coefficients in each time period with the previous ones.
Below are some characteristics of environments to recognize them:

1. For indoors the difference between the coefficients is so high in comparison with scale of them in the first group. This way we can distinguish between indoor and outdoor.
2. For unstructured areas it will happen for the second group, and through this information we can find out that we are in a structured or unstructured area.
3. In dynamic areas changing of the third and fourth groups data is so high, and it can help us to recognize the static and dynamic areas.
4. In large scale areas the obstacle detecting sensors will find less objects then the small scale.

Indeed, the parts explained above are just a part of our algorithm to find out the characteristics of the environment. And after modifying the coefficients the role of one method may pales a lot.

5.3. Navigation Methods Implementation

5.3.1. Navigation Group I

This group uses absolute positioning method and contains a GPS. One powerful microcontroller captures the GPS information which contain longitude and latitude and other information and after transformation to Cartesian coordinate and calculation of reliability cofactor according to special algorithms, sends them to the main controller. Then on the Base Times main controller sends this information to main processor. This results in reduce wiring and better timing.

This group has the most effect on localization on outdoor and large scale environments. A minimum of four satellites have to be detected by the receiver to give a position estimate, with the more satellites detected, the more accurate the position estimate. The position is calculated through a trilateration technique based on the TOF information.

5.3.1.1. GPS Deficiencies

The errors in an uncorrected GPS signal come in many forms and arise from a variety of different sources. These errors have been divided into two broad categories: 1) high frequency noise and 2) long-term drift. The first category pertains to the errors that manifest themselves as high frequency noise or spikes. These errors are easily identifiable on a 2-D plot of the GPS track recorded from a moving platform. Although, no attempt has been made to formulate an explicit definition of what constitutes noise, a general example would be single-epoch jumps in the GPS position. An epoch is one GPS cycle (milliseconds). The difficulty arises from the fact that in some instances the position can jump several meters and then either jump back on the next epoch or maintain that new position for a few seconds or indefinitely. If the new position is maintained for more than approximately 30 seconds, then it is no longer considered noise but lies in the gray area between the two categories.

Experience has shown that the two main causes of GPS noise are satellites coming in and out of the view of the GPS receiver and multi-path effects. The magnitude of these errors varies from a few feet to hundreds of feet.

The second category of GPS error is classified as drift. These errors are much more difficult to see on a track plot, since they change over a period of hours rather than seconds like the noise errors. It is difficult to determine the exact cause of these types of errors, but they are typically attributed to atmospheric effects in the ionosphere and troposphere and satellite geometry. The magnitude of these errors can vary from no error at all to thirty feet or more.

5.3.1.2 GPS noise remedy

In order to perform reasonable waypoint navigation, a robot needs to have a relatively noise-free estimate of its current state. Obviously, a non-differential GPS solution alone is not capable of providing that estimate.

The most common solution for solving the problem of GPS noise (and the solution used here) is to augment the GPS with other sensors and employ a Kalman Filter to optimally combine all of those sensor inputs. An inertial sensor is an ideal companion for the GPS in a navigation package, as the two sensors have complementary errors (i.e. inertial sensors generally have very little noise but drift without limit, whereas, GPS is quite noisy but has finite drift). By using Kalman Filter almost all the spikes in the GPS noise have been smoothed out. The Kalman Filter does an excellent job of compensating for the noise in the GPS position, but is of no help with the long-term drift error in the GPS position.

5.3.2. Navigation Group II

This group takes advantage of combining of two methods: inertial and odometry. This group contains a electrical compass, a 3DM sensor (compose of 3 accelerators, 3 gyroscopes, 3 magnetometer), tachometer (shaft encoder) and feedback of motor current and torque. We combine these two methods in one group to solve the problem of their errors in low level.

One powerful microcontroller (processor) reads information of all sensors and combines them together, according a special algorithm to obtain the real momentum. With having the spatial direction of movement and real momentum we could obtain the position of robot and send them to the main control on each Base time. Then on the Base Times main controller sends this information to main processor. This results in reduce wiring and better timing. Problem of this method is accumulative error that increases. We solve this problem by updating the group in the end of each Base Time with sending information by main processor

5.3.2.1. Electrical Compass

Vehicle heading is the most significant of the navigation parameters (x, y, and θ) in terms of its influence on accumulated dead-reckoning errors. For this reason, sensors which provide a measure of absolute heading are extremely important in solving the navigation needs of autonomous platforms. The magnetic compass is such a sensor. One disadvantage of any magnetic compass, however, is that the earth's magnetic field is often distorted near power lines or steel structures. This makes the straightforward use of geomagnetic sensors difficult for indoor applications but it is suitable for outdoor and large scale environment. Based on a variety of physical effects related to the earth's magnetic field, different sensor systems are available:

- Mechanical magnetic compasses.
- Fluxgate compasses.
- Hall-effect compasses.
- Magnetoresistive compasses.
- Magnetoelastic compasses.

The compass best suited for use with mobile robot applications is the fluxgate compass. When maintained in a level attitude, the fluxgate compass will measure the horizontal component of the earth's magnetic field, with the decided advantages of low power consumption, no moving parts, intolerance to shock and vibration, rapid start-up, and relatively low cost. If the vehicle is expected to operate over uneven

terrain, the sensor coil should be gimbal-mounted and mechanically dampened to prevent serious errors introduced by the vertical component of the geomagnetic field.

5.3.2.2. 3DM

As mention above 3DM contains compose of 3 accelerators, 3 gyroscopes and 3 magnetometer and it specifies the spatial orientation of robot very accurately. In fact by using of 3DM we have the direction of movement of robot in spatial space.
Note that 3DM and compass are complementary and cooperate with each other to determine the direction of the robot.
3DM Deficiency is static angle error that can be removed by other mechanism.

5.3.2.3. Tachometer & Motor Current

Tachometer (optic shaft encoder) determines the rotation of the motor shaft but because of slipping it is not exactly the real momentum of the robot. We use the feedback of motor current and current to determine approximately the momentum of the robot. This is done by a special algorithm that compares the real motor current to the value of the nominal motor current in that motor rate and specifies approximately the slipping value.

5.3.2.4. Minimizing Odometry Error

Error sources fit into one of two categories: (1) systematic errors and (2) non-systematic errors:

1. Systematic errors
a. Unequal wheel diameters
b. Average of both wheel diameters differs from nominal diameter
c. Misalignment of wheels
d. Uncertainty about the effective wheelbase (due to non-point wheel contact with the floor)
e. Limited encoder resolution
f. Limited encoder sampling rate

2. Non-systematic errors
a. Travel over uneven floors
b. Travel over unexpected objects on the floor
c. Wheel-slippage due to:
- Slippery floors
- Over-acceleration
- Fast turning (skidding)
- External forces (interaction with external bodies)
- Internal forces (e.g., castor wheels)

- Non-point wheel contact with the floor

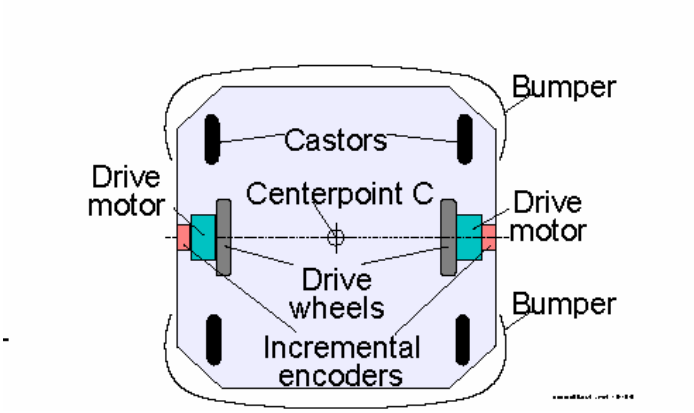


Fig. 26 A typical differential-drive mobile robot (bottom view)

Systematic errors are particularly grave, because they accumulate constantly. On most smooth indoor surfaces systematic errors contribute much more to odometry errors than non-systematic errors. However, on rough surfaces with significant irregularities, non-systematic errors may be dominant.
We do many efforts due to reducing both of systematic and nonsystematic errors.

5.3.2.4.1. Measurement of Systematic Errors
One important but rarely addressed difficulty in mobile robotics is the *quantitative* measurement of odometry errors. Lack of well-defined measuring procedures for the quantification of odometry errors results in the poor calibration of mobile platforms and incomparable reports on odometric accuracy in scientific communications. To overcome this problem Borenstein and Feng [1995] developed a method for quantitatively measuring systematic odometry errors and, to a limited degree, non-systematic odometry errors. This method, called *University of Michigan Benchmark* (UMBmark) requires that the mobile robot be programmed to follow a preprogrammed square path of 4x4 m side-length and four on-the-spot 90-degree turns. This run is to be performed five times in clockwise (cw) and five times in counter-clockwise (ccw) direction.
When the return position of the robot as computed by odometry is compared to the actual return position, an error plot similar to the one shown in Figure 1 will result. The results of Figure 1 can be interpreted as follows:

- The stopping positions after cw and ccw runs are clustered in two distinct areas.
- The distribution within the cw and ccw clusters is the result of non-systematic errors. However, Figure 1 shows that in an uncalibrated vehicle, traveling over a reasonably smooth concrete floor, the contribution of *systematic* errors to the total odometry error can be notably larger than the contribution of non-systematic errors.

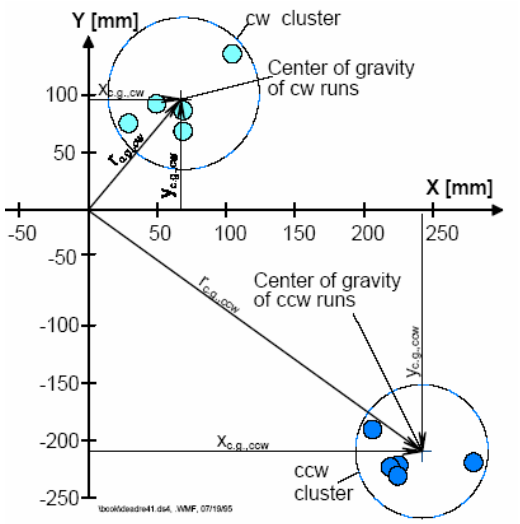


Fig. 27 Typical results from running UMBmark (a square path run five times in cw and five times in ccw directions) with an uncalibrated TRC LabMate robot.

The asymmetry of the centers of gravity in cw and ccw results from the dominance of two types of systematic errors, collectively called Type A and Type B [Borenstein and Feng, 1996]. Type A errors are defined as orientation errors that reduce (or increase) the amount of rotation of the robot during the square path experiment in ***both*** cw and ccw direction. By contrast, Type B errors reduce (or increase) the amount of rotation when traveling in cw but have the opposite effect when traveling in ccw direction. One typical source for Type A errors is the uncertainty about the effective wheelbase; a typical source for Type B errors is unequal wheel diameters.
After conducting the UMBmark experiment a single numeric value that expresses the odometric accuracy (with respect to systematic errors) of the tested vehicle can be found from:

$$E_{\max,\text{syst}} = \max(r_{\text{c.g.,cw}} ; r_{\text{c.g.,ccw}}) . \quad (1)$$

where

$$r_{c.g.,cw} = \sqrt{(x_{c.g.,cw})^2 + (y_{c.g.,cw})^2}$$

and

$$r_{c.g.,ccw} = \sqrt{(x_{c.g.,ccw})^2 + (y_{c.g.,ccw})^2} .$$

Based on the UMBmark test, Borenstein and Feng [1995; 1996] developed a calibration procedure for reducing systematic odometry errors in differential drive vehicles. In this procedure the UMBmark test is performed five times in cw and ccw direction

to find $x_{c.g.cw}$ and $x_{c.g.ccw}$. From a set of equations defined in [Borenstein and Feng two calibration constants are found that can be included in the basic odometry computation of the robot. Application of this procedure to several differential-drive platforms resulted consistently in a 10- to 20-fold reduction in systematic errors. Figure 2 shows the result of a typical calibration session. E_{maxsys} The results for many runs calibration sessions with TRC's LabMate robots averaged E_{maxsys} = 330 mm for uncalibrated vehicles and E_{maxsys} = 24 mm after calibration.

5.3.2.4.2. Measurement of Non-Systematic Errors

Borenstein and Feng also propose a method for measuring non-systematic errors. This method, called *extended UMBmark,* can be used for comparison of different robots under similar conditions, although the measurement of non-systematic errors is less useful because it depends strongly on the floor characteristics. However, using a set of well-defined floor irregularities and the UMBmark procedure, the *susceptibility* of a differential-drive platform to nonsystematic errors can be expressed. Experimental results from six different vehicles, which were tested for their susceptibility to *non-systematic* error by means of the *extended UMBmark* test, are presented in Borenstein and Feng.

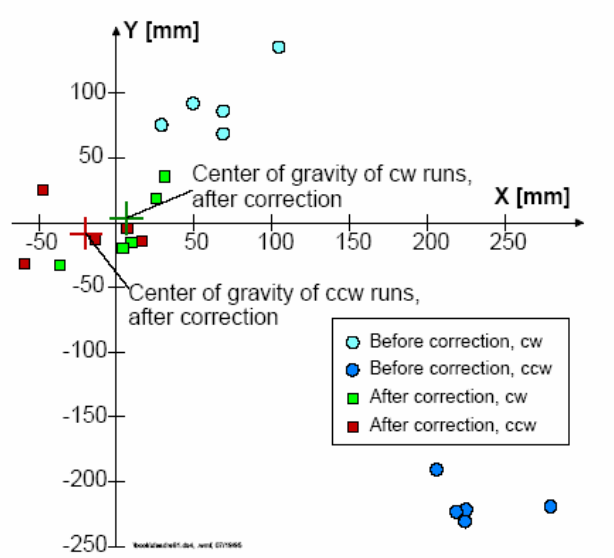


Fig. 28 Position errors after completion of the bidirectionalsquare-path experiment (4 x 4 m).

Borenstein developed a method for detecting and rejecting non-systematic odometry errors in mobile robots. With this method, two collaborating platforms continuously and mutually correct their nonsystematic (and certain systematic) odometry errors, even while both platforms are in motion. A video entitled "CLAPPER" showing this system in operation is included in [Borenstein et al., 1996b]) and in [Borenstein

1995v]). A commercial version of this robot, shown in Figure 3, is now available from [TRC] under the name "*OmniMate.*" Because of its internal odometry error correc tion, the OmniMate is almost completely insensitive to bumps, cracks, or other irregularities on the floor.

5.3.3. Navigation Group III

This group uses ranging method and contains 4 IR ranger and 6 ultrasonic sensors. We take advantage of a new sonar based localization method suitable for both static and dynamic environments. Instead of local grids, the system stores a short feature vector, which is obtained from sonar readings by means of a simple transformation. The proposed vectors present the following advantages: 1) they can be created at any position, even in unstructured environments; 2) they do not depend on the robot orientation.

Data capturing and basic calculation is done by a powerful microcontroller and after determining the reliability cofactor send information to the main controller. Then on the Base Times main controller sends this information to main processor. This results in reduce wiring and better timing.

5.3.3.1. Place learning algorithm

A sonar reading typically provides information about the distance of the sensor to the closest obstacle in the direction of the beam. Since sonars present an arc of uncertainty, most systems rely on accumulating evidence from several ones, which are usually integrated into an evidence grid . If a mobile robot builds an evidence grid while it knows its correct position, it can compare local grids adquired after losing its reference to the global one to locate itself.

In order to achieve a grid accurate enough to represent most significant features within a typical indoor environment, such a grid must present an adequate decomposition degree and, therefore, it may yield a relatively large data volume. To reduce the problem instance, many methods rely on searching significant structures over the adquired evidence grids. However, they work with very simple ones, like walls or corners, and, therefore, the approach is not efficient for complex unstructured environments. Besides, the robot can only locate itself if it is close to any of these structures and if their layout is significant enough to be distinguished (i.e. if the robot only recognizes walls, it can not locate itself in a square room, even if there are chairs and tables around).

Instead of searching for simple structures, we propose a method to transform multiple sonar readings into a small vector which can efficiently represent any point of the environment. If a robot is equipped with a ring of sonar sensors, an evidence grid basically provides information about the distance between the robot and the closest obstacles around for each beam direction in polar coordinates. This information can be stored into a one-dimensional function known as depth map, which represents the contour of the area free of obstacles around the robot. The similarity of two depth maps can be calculated by means of a circular correlation to avoid dependence on the orientation on the robot. Clearly, depth maps involve less data than an average evidence grid and, being one-dimensional, they are easier to compare. A complete depth

map yields 360 points, but it can be subsampled to lower lengths. Fig. 1.b presents a 64-points depth function adquired at a random position of a cluttered environment.

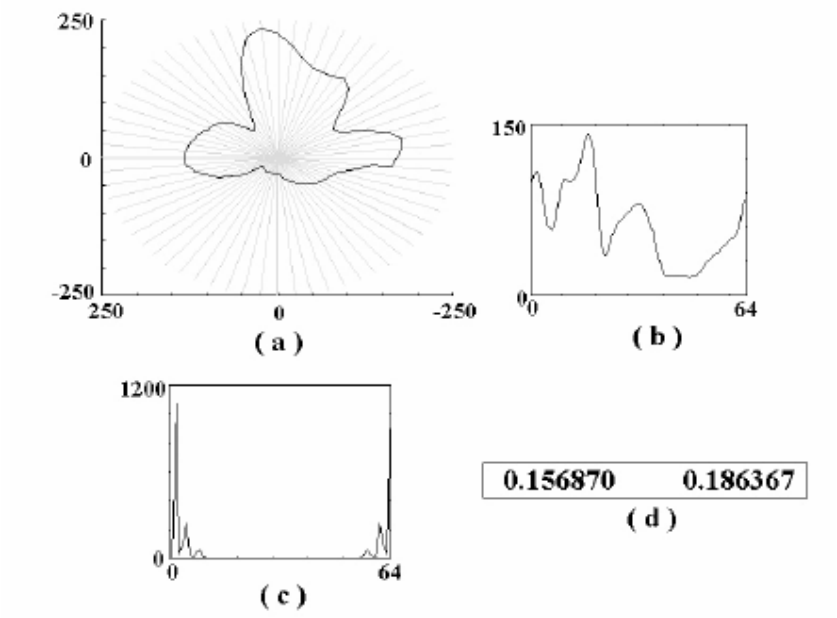


Fig. 29 Landmark adquisition: a) sonar reading; b)depth map; c) DMDFT; d) landmark

Depth maps are still too large to become suitable landmarks, but it can be observed that the Discrete Fourier Transform of an average depth map (DMDFT) yields a large number of zero components (Fig. 1.c). Since the Fourier space for FFTs of N points is an N-dimensional vectorial space, it can be assumed that the DMDFTs conform a P-dimensional vectorial subspace, being P lowers than N. Thus, the shortest vector capable of uniquely representing a DMDFT is equal to P, and it is calculated by projecting the DMDFT onto a base of the subspace it belongs to.

Calculation of a base is not simple because some non-zero components of the DMDFT could be linearly dependant. Since a base of a P-dimensional subspace is a set of P orthogonal vectors, we propose a simple approach to calculate one. The method consists of clustering a set of random DMDFTs by means of a k-means algorithm relying on euclidean distance. A k-means algorithm [3] split the sample space into k classes whose prototypes - which are equal to the average of all the elements of the class - tend to be as different as possible. Thus, if k is lower or equal to P, prototypes are as orthogonal as possible. Obviously, P is not known a priori, but it can be easily calculated by clustering the DMDFTs for increasing values of k: if k is larger than P, the average angle between prototypes decreases because there can be no more than P different orthogonal vectors in a P-dimensional subspace. The prototypes of

the classes for k=P are the vectors of the base and the coordinates of a DMDFT onto the base are calculated as:

$$\alpha_i = \sum_N m_j v_{ij} \quad \forall i \leq P \qquad (1)$$

αi being the i-th coordinate of the DMDFT in the new base, mj the j-th component of the DMDFT and vij the j-th component of the i-th vector of the base. These vectors are not really orthogonal and there exists a representation error, but it has experimentally proven to be very small.

To prove that the number of samples required calculating a base does not necessary need to be large in this work only 30 depth maps adquired at random positions of a structured simulated environment were used. All tests were performed nevertheless in a unstructured real environment to prove that the base is valid for environments different from those were it was calculated. In order to estimate the correct dimension of the subspace, several clustering processes were performed for ks ranging from 2 to 7. The best results were achieved for a k equal to 2 (Fig. 2), when the average angle between vectors was maximum (67.56 degrees). Despite the lack of orthogonality of the vectors of the base, it can be observed in Fig. 3 that closes places yield very similar landmarks and different ones do not. Obviously, if two different places yield a similar layout, their landmarks will be similar as well (landmarks 7-9 and 1-8).

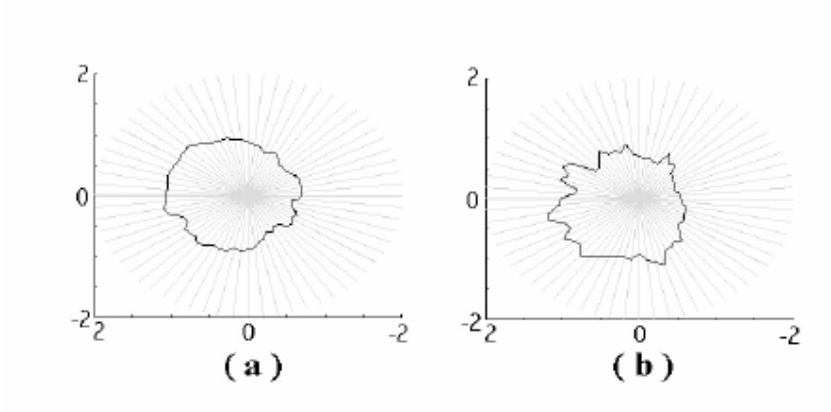


Fig. 30 A base of the vectorial subspace of DMDFTs: a) vector 1; b) vector 2.

After off-line calculation of a base, a landmark is adquired on-line through three steps:

-Adquisition of a depth map from the sonar readings

- Calculation of the Fourier Transform of the map (DMDFT)

-Projection of the DMDFT onto the vectors of the base

It must be observed that the last step simply consists of P products of N-dimensional vectors. Thus, it is very fast and it does not depend on the local complexity of the

environment. Also, the data volume required to represent any place is constant, despite its particular features, and therefore place recognition is easier.

5.3.3.2. Place recognition algorithm

5.3.3.2.1. Static localization

Static localization typically consists of comparing the current landmark to all stored ones. The robot is presumedly located at the position whose landmark is most similar to the current one. Obviously, if landmarks are adquired as often as possible, the environment can be constantly actualised and the robot can locate itself at more places. However, processing time increases with the number of stored landmarks. Since close landmarks tend to be very similar, this problem can be partially solved by segmenting stored landmarks into regions by means of any algorithm relying both on spatial proximity and similarity between landmarks. This process divides space into compact regions yielding similar landmarks and each region is represented by a prototype, which is equal to the average of all the landmarks of the region. Although landmarks are not usually available at every point inside a region, it is interpolated that they all present the prototypical one. Segmentation presents the following advantages: i) comparing a landmark to available prototypes is faster than comparing it to all stored landmarks; ii) prototypes are not as affected by transient changes and spurious reflections as individual landmarks; and iii) all explored environment present a landmark value even if no landmark was adquired there.

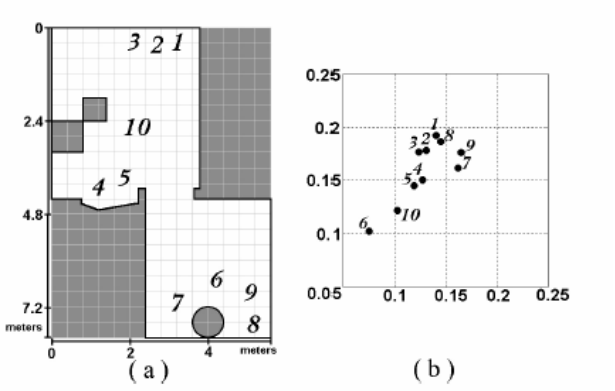


Fig. 31 a) real environment and landmark adquisition points; b) 2-dimensional landmarks at points 1-10

5.3.3.2.2. Dynamic localization

After the landmark map is segmented the robot may perform a static localization process, but this approach still presents two problems: 1) several places could present very similar or equal landmarks; and 2) the landmark adquired by the robot to locate itself may be distorted by transient or lasting changes in the environment. These problems may be solved by accumulating evidence of being at a given region through several movements. This process is typically performed by means of Markov chains [1], which are stocastic processes where the probability of reaching a future state Rj

8j > n depends only on the present state Rn. Formally, for any succession of states r1, r2, ...rn+1:

$$Pr(R_{n+1} = r_{n+1} | R_1 = r_1, ... R_n = r_n) = Pr(R_{n+1} = r_{n+1} | R_n = r_n) \quad (2)$$

Localization presents a finite number of states N which is equal to the number of different regions the robot may be located at. The probability of being at region rj at instant n + 1 if the robot was at region ri at instant n (Pr(Rn+1 = rj jRn = ri)) is known as probability of transition, tij . Given a finite Markov chain of N states, these probabilities of transition tij conform a matrix T of NxN elements, being tij≥0 and

$$\sum_{j=1}^{N} t_{ij} = 1 \text{ for } i = 1, ...N$$

Obviously, T depends on the direction and magnitude of the robot movement, which are constants for the transition matrix. However, sometimes a robot may find an obstacle in its direction. In this case, it is desirable to calculate more matrices so that the robot can continue the localization process.

The calculation of a transition matrix at direction d (Dd) consists of the following steps:

-For each point of a region i of the localization map, it is evaluated to which region the robot arrives after a movement of magnitude M and direction d. If the arrival region is j, let Dd{i,j}= Dd{i,j}+1;

-Normalization of the matrix by factor

$$\sum_{i=1}^{N} D\{i, j\}.$$

5.3.3.2.3. Hybrid localization

After an initial static estimation, localization may rely uniquely on Markovian chains, but the efficiency of the process depends on the correctness of the initial estimation. However, there are usually several locations whose landmark is similar to the first adquired one. Also, the robot could be initially located in an unexplored area, the environment might have changed or the first landmark could be distorted. In all cases, it is necessary to correct the initial uncertainty.

This paper proposes to merge static and dynamic localization techniques: each time the robot moves, its probability of being at each region is partially given by Markov chains, but also by the likeness of the most recently adquired landmark to the prototypes of existing regions. In order to combine data, two different probability vectors are defined: the static vector ***v***, which relies uniquely on current sonar readings, and the dynamic vector ***w***, which relies on Markov chains. After transition matrices have been calculated, the dynamic vector at instant n is easily obtained as:

$$\vec{w_n} = \vec{x}_{n-1} D \quad (3)$$

(D) being the transition matrix in the direction of the movement and xn-1 the combined probability vector in the previous instant. In order to calculate the static vector, the euclidean distance of the current landmark to all available prototypes is studied: the larger the distance to a given prototype, the less likely it is that the robot occupies the region. Thus, the static vector at moment n can be obtained by simple normalization:

$$v_n(i) = \frac{1 - d(i) / \sum_{i=1}^{N} d(i)}{\sum_{i=1}^{N} d(i)} \qquad (4)$$

vn(i) being the i-th component of the static vector and d(i) being the euclidean distance between current landmark and class i prototype. Thus, given an agent moving in a partially explored environment segmented into N classes, the localization algorithm consists of the following steps:

1. Calculation of the transition matrices.
2. Calculation of the initial static vector of probabilities **v**o.
3. Let the initial vector of combined probability **x**0 = **v**0. Let n = 0.
4. Estimation of the possible location of the agent (Rloc), which is the set of regions whose probability of occupation xn[i] is higher than a threshold Uloc. If Rloc is connected and its area is small enough to provide the required precision, the process finishes.
5. Let n = n + 1.
6. Movement of magnitude M in any free direction depending on the defined matrices.
7. Calculation of the new static vector of probabilities **v**n.

$$\vec{w_n} = \vec{x}_{n-1} D$$

where D is the transition matrix corresponding to the chosen movement direction.

8. Calculation of the new dynamic vector of probabilities
9. Calculation of the new vector of combined probability **x**n as:

$$\vec{x_n} = \frac{\gamma \vec{v_n} + \vec{w_n}}{\gamma + 1} \qquad (5)$$

γ being a heuristically estimated pondering factor. The robot may be located at any region i whose xi is larger than a threshold.

10. Go to step 4 until there is a single region the robot may be located at.

It must be noted that this algorithm is valid as long as landmarks can be adquired at any position of the environment.

5.3.4. Navigation Group IV

This group also uses ranging method and contains a laser line scanner. The information of the laser scanner is processed by main processor. The laser scanner can provide an instance 2D map that contains direct distance between robot and obstacles. For the best performance we use a algorithm that its theory mentions below:

Motion estimation from range imagery takes part in three stages: terrain map generation, terrain map alignment and motion estimation.

5.3.4.1. Terrain Map Generation

Terrain map generation is the process by which range samples are projected into a grid to form a 2½-D terrain map representation. Scanning laser rangefinders generally

have spherical or perspective projection models. Also, scan patterns are not always regular raster scans; spiral and helical scans are common when minimizing scanner power. Nonlinear projection models and irregular scan patterns create an irregular sampling of the surface. If the range samples are used directly, a time consuming registration algorithm that accounts for the irregular spacing between samples is needed (e.g., ICP). However, by resampling the range samples from each scan to a regular grid in Cartesian space, motion estimation can be posed as an image alignment problem greatly simplifying the underlying algorithms and data structures, which will ultimately result in a more efficient algorithm. A terrain map is a function *Z(r,c)* that encodes elevation on a regular grid. To generate a terrain map, the horizontal size of each grid cell, *s,* and horizontal extent, *h*, of the terrain map must be determined. As shown in Figure 1, these parameters can be determined from the scanner field of view *f*, the average of scan samples across the scene *n*, and the average range to the scene being imaged *R*. In general we set these parameters as follows:

$$H = 2R\tan(f/2) \qquad s = h/n$$

With these settings, the terrain map will cover roughly the same extent as the scanned data and each grid cell will contain approximately one sample.

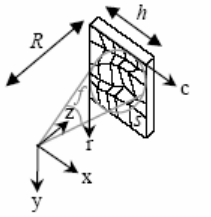


Fig. 32 Sensor and terrain map coordinates.

Once the terrain map parameters are established, the procedure for terrain map generation is as follows. First, each range sample is converted from scanner angle and range coordinates to Cartesian coordinates *(x,y,z)*. Next, the *(x,y)* coordinates of the sample are used to determine the floating point coordinates *(r,c)* that the sample projects to in the grid cell

$$(r,c) = (y/s + h/2, x/s + h/2)$$

The coordinate relationship between sensor and terrain map coordinates is shown in Figure 1. In general *(r,c)* will fall between discrete grid cells, so, to prevent aliasing, bilinear interpolation is used to update the terrain map.

Two arrays are used to perform bilinear interpolation: the elevation accumulator *E(r,c)* and the bilinear weight accumulator *W(r,c)*. For each sample, the four grid cells surrounding *(r,c*) are updated using

$$p = r - \underline{r} \quad q = c - \underline{c}$$
$$E(r,c) += (1-p)(1-q)z \quad W(r,c) += (1-p)(1-q)$$
$$E(r+1,c) += p(1-q)z \quad W(r,c) += p(1-q)$$
$$E(r,c+1) += (1-p)qz \quad W(r,c) += (1-p)q$$
$$E(r+1,c+1) += pqz \quad W(r,c) += pq$$

Where x is the floor operator. After all samples have been accumulated, the elevation Z at each grid cell is determined using

$$Z(r,c) = E(r,c) / W(r,c)$$

Due to the irregular sampling by the scanner, it is possible that a grid cell did not have a sample projected into it and consequently does not have an elevation value. For efficiency during image alignment, it is important that the terrain map be free of holes, especially near the center of the map. A simple interpolation scheme is used to fill any holes. First, hole cells are detected using a modified grassfire transform that detects cells that do not have an elevation but are surrounded by cells with elevation. Next, each hole cell is assigned the average elevation of all neighboring cells that have elevation values. By repeating this process until all hole cells have an elevation value, the holes in the terrain map are filled incrementally. Figure 2 shows a typical range scan, a terrain map before hole filling and a terrain map after hole filling.

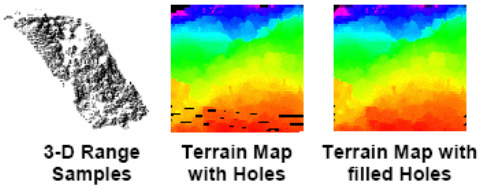


Fig. 33 Terrain map generation.

To be aligned by our algorithm, two terrain maps must be generated using the same terrain map parameters. Also any rotation between the scans must be eliminated before the scans are aligned. The following procedure is used to generate two terrain maps for alignment. First, the range samples in each scan are converted to Cartesian coordinates. Next, the rotation between the scans (determined from on board gyros) is eliminated by rotating the samples from the second scan into the frame of the first scan. Next, the terrain map parameters are determined using the data from the first scan. These parameters are then used to generate the terrain maps for both scans ensuring that the sizes of the grid cells are the same for each image. The end results of terrain map generation are two terrain maps that are ready for terrain map alignment.

5.3.4.2. Terrain Map Alignment

During terrain map alignment one terrain map is shifted relative to another by **d** = *(dr,dc,dz)* until the difference in elevation data between the two maps is minimized. Our procedure for terrain map alignment is inspired by the Shi-Tomasi feature tracker [7]. However, we modify the tracker to use the additional elevation information to provide full 3-D tracking. Suppose the terrain map *I(r,c)* is generated and then, using samples from a later scan, the terrain map *J(r,c)* is generated. We would like to solve for the 3-D shift **d** between the scans. Following the derivation of Shi and Tomasi, at the correct shift, the relationship

$$I(r+dr, c+dc) + dz = J(r,c)$$

holds. To constrain the problem so that we can solve for the 3-D shift and account for noise in the data, we seek to minimize

$$\varepsilon = \sum_{W} (I(r+dr, c+dc) + dz - J(r,c))^2 \qquad (1)$$

over a window W that covers most of the terrain map. The minimum of e can be found by differentiating e with respect to the image shift **d** and setting the result to zero

$$\frac{1}{2}\frac{\partial \varepsilon}{\partial d} = \sum_{W} (I(r+dr, c+dc) + dz - J(r,c))\mathbf{g} = 0 \qquad (2)$$

Where

$$\mathbf{g} = (\partial I/\partial r, \partial I/\partial c, 1)\,.$$

Finite differences are used to compute the gradients of the terrain map

$$\frac{\partial I}{\partial r} = \frac{I(r+1,c) - I(r-1,c)}{2} \quad \frac{\partial I}{\partial c} = \frac{I(r,c+1) - I(r,c-1)}{2}$$

If the image shift is small then $I(r+dr,c+dc)+dz$ can be approximated by its truncated Taylor series expansion

$$I(r+dr, c+dc) + dz = I(r,c) + \mathbf{g}^T \mathbf{d} \qquad (3)$$

Substituting (3) into (2) and rearranging terms results in

$$\sum_{W} (J(r,c) - I(r,c))\mathbf{g} = (\sum_{W} \mathbf{g}\mathbf{g}^T)\mathbf{d}$$

This is a linear equation in the unknown **d**

$$\mathbf{e} = H\mathbf{d} \qquad (4)$$

$$\mathbf{e} = \sum_{W} (J(r,c) - I(r,c)\mathbf{g} \quad H = \sum_{W} \mathbf{g}\mathbf{g}^T$$

Because of the linearization, the solution to (4) does not minimize (1) exactly. However using (4), a Newton Raphson iterative minimization can be used to align the terrain maps exactly. The procedure is to first solve (4) for **d0** (H and **e** are constructed assuming **d** = 0). Then iteratively solve (4) for **di** with **e** replaced by

$$\mathbf{e_i} = \sum_{W} (J(r,c) - I(r + dr_i, c + dc_i) - dz)\mathbf{g}$$

until **di** changes very little. **di** is a floating point value, so $I(r+dri,c+dci)$ is determined through bilinear interpolation of the four neighboring grid cells to $(r+dri,c+dci)$. The end result of terrain map alignment is a vector **d** that aligns the two terrain maps.

The window W over which the two terrain maps share data, and therefore can be compared, changes at each iteration. It is possible to determine W at each iteration so that all possible data is used. However, if W is fixed for all iterations, a more efficient algorithm results because H is computed only once per alignment.

In order to maximize the overlap between terrain maps and consequently minimize the alignment error, W should be set as large as possible. However, because of boundary effects, it is not possible to set W to the entire terrain map. Ideally, W will be set such that when the terrain maps are aligned using the correct transformation (rotation and translation) W is the largest window contained completely within both maps. Since it is not possible to know the translation between terrain maps before

alignment, our algorithm sets W by using a translation extrapolated from the translation computed between the previous two scans. Another alternative is to set W based on a translation predicted from on-board inertial sensors.

5.3.4..3 Motion Estimation

The purpose of motion estimation is to transform the alignment vector **d** into a 3-D translation **T** and also compute the covariance matrix C of the translation. **T** can be computed directly from **d** by

$$\mathbf{T} = S\mathbf{d}$$

Where

$$S = \begin{bmatrix} 0 & s & 0 \\ s & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Since d is estimated using least squares, the covariance of **d** is the inverse of H. Given that **T** is a linear function **d**, the covariance of **T** can be computed from H as well.

$$C = (S^T HS)^{-1}\sigma^2$$

σ^2 is the variance on the terrain map noise which can be computed from sensor noise characteristics. Once the translation and translation covariance are computed, they can be passed to the spacecraft guidance, navigation and control subsystem for execution of safe and precise trajectories.

5.3.4.4. Multi-frame Motion Estimation

During autonomous landing, multiple range scans will be taken as the lander approaches the comet surface. For small translations, alignment errors are roughly independent of the magnitude of translation; in general they are between 1/3 and 1/6 the terrain map grid cell size. If motion estimation is done between subsequent scans, then the motion estimation error will accumulate a fixed error for each scan. However, if the translation between scans is small with respect to the extent of the surface area scanned, then it will be possible to align multiple range scans to a single key scan. In this case, the alignment errors will remain fixed for each key scan resulting in a less rapid growth in alignment errors. At some point, it will become difficult to align a scan with the current key scan because the overlap becomes too small. When this happens, the key scan is updated to the current scan. Although the accumulation of errors cannot be eliminated, this procedure will keep it to a minimum.

Using key scans also has advantages in terms of efficiency. During alignment, terrain map gradients are computed only for the first map. Since the first map corresponds to the key scan, gradients will only have to be computed each time the key frame is changed. Since computing image gradients takes roughly half the total time to estimate motion between two scans, eliminating this step results in an algorithm that is twice as fast.

Deciding when to select a new key scan is not straight forward. This decision depends on the surface overlap between scans and the overall roughness of the surface being scanned. In our algorithm, we select a new key frame based solely on the overlap

between scans; when the window of overlap *W* between scans falls below a threshold based on the number of grid cells in the terrain maps, a new key frame is selected.
Now that we have discussed all of the components, we can describe our motion estimation algorithm in its entirety. The first scan is taken, its terrain map is generated and the gradients of this terrain map are computed. This is the key scan. The next scan is taken and its terrain map is generated using the terrain map parameters of the key scan. The comparison window *W* is set based on an initial prediction of the translation between scans. Next the terrain maps are aligned and the motion and the motion covariance between scans are computed. The next scan is read in, and its terrain map is generated using the parameters of the key scan. *W* is set based on the motion extrapolated from previous alignments. The current terrain map and the key map are aligned. This procedure repeats until *W* shrinks below ½ the total number of grid cells in the terrain maps; at this point a new key map is selected and the procedure repeats.

5.3.5. Navigation Group V

This group takes advantage of image processing techniques and contains a panoramic camera. This camera provides a 360 degree view of environment that would be very useful for mapping. Images of this camera directly by main processor with assistance of a FPGA board have processed.

Virtually all existing localization algorithms extract a small set of *features* from the robot's sensor measurements. *Landmark-based approaches*, which have become very popular in recent years, scan sensor readings for the presence or absence of landmarks to infer a robot's position. Other techniques, such as most *model matching approaches*, extract certain geometric features such as walls or obstacle configurations from the sensor readings, which are then matched to models of the robot's environment. The range of features used by different approaches to mobile robot localization is quite broad. They range from artificial markers such as barcodes and more natural objects such as ceiling lights and doors to geometric features such as straight wall segments and corners. This raises the question as to what features might be the best ones to extract, in the sense that they produce the best localization results.
The problem of learning the right landmarks has been recognized as a significant scientific problem in robotics (Borenstein, Everett, & Feng, 1996), artificial intelligence (Greiner & Isukapalli, 1994), and in cognitive science (Chown, Kaplan, & Kortenkamp, 1995).
Few localization algorithms enable a robot to learn features or to define its own landmarks. Instead, they rely on static, handcoded sets of features for localization, which has three principle disadvantages:

1. **Lack of flexibility.** The usefulness of a specific feature depends on the particular environment the robot operates in and also often hinges on the availability of a particular type of sensors. For example, the landmark "ceiling light"—which has been used successfully in several mobile robot applications—is useless when the environment does not possess ceiling lights, or when the robot is not equipped with the appropriate sensor (such as a camera). If the features are static and predetermined, the robot can localize itself only in environments where those features are meaningful, and with sensors that carry enough information for extracting them.

2. **Lack of optimality.** Even if a feature is generally applicable, it is usually unclear how good it is or what the *optimal* landmark would be. Of course, the goodness of features depends, among other things, on the environment the robot operates in and the type of uncertainty it faces. Existing approaches usually do not strive for optimality, which can lead to brittle behavior.

3. **Lack of autonomy.** For a human expert to select appropriate features, he/she has to be knowledgeable about the characteristics of the robot's sensors and its environment. Consequently, it is often not straightforward to adjust an existing localization approach to new sensors or to new environments. Additionally, humans might be fooled by introspection. Since the human sensory apparatus differs from that of mobile robots, features that appear appropriate for human orientation are not necessarily appropriate for robots.

We present an algorithm, called BaLL (short for *Bayesian landmark learning*), that lets a robot *learn* such features, along with routines for extracting them from sensory data. Features are computed by artificial neural networks that map sensor data to a lowerdimensional feature space. A rigorous Bayesian analysis of probabilistic mobile robot localization quantifies the average posterior error a robot is expected to make, which depends on the features extracted from the sensor data. By training the networks so as to minimize this error, the robot learns features that directly minimize the quantity of interest in mobile robot localization (see also Greiner & Isukapalli, 1994).

We conjecture that the learning approach proposed here is more *flexible* than static approaches to mobile robot localization, since BaLL can automatically adapt to the particular environment, the robot, and its sensors. We also conjecture that BaLL will often yield *better results* than static approaches, since it directly chooses features by optimizing their utility for localization. Finally, BaLL increases the *autonomy* of a robot, since it requires no human to choose the appropriate features; instead, the robot does this by itself. The first and the third conjecture follow from the generality of the learning approach. The second conjecture is backed with experimental results which illustrate that BaLL yields significantly better results than two other approaches to localization.

5.3.5.1. A probabilistic model of mobile robot localization

This section lays the groundwork for the learning approach presented in next Section, providing a rigorous probabilistic account on mobile robot localization. In a nutshell, probabilistic localization alternates two steps:

1. **Sensing.** At regular intervals, the robot queries its sensors. The results of these queries are used to refine the robot's internal belief as to where in the world it is located. Sensing usually decreases the robot's uncertainty.
2. **Acting.** When the robot executes an action command, its internal belief is updated accordingly. Since robot motion is inaccurate due to slippage and drift, it increases the robot's uncertainty.

The derivation of the probabilistic model relies on the assumption that the robot operates in a partially observable Markov environment (Chung, 1960) in which the only "state" is the location of the robot. In other words, the Markov assumption states that

noise in perception and control is independent of noise at previous points in time. Various other researchers, however, have demonstrated empirically that the probabilistic approach works well even in dynamic and populated environments, due to the robustness of the underlying probabilistic representation.

5.3.5.1.1. Robot motion

BaLL employs a probabilistic model of robot motion. Let denote the location of the robot within a global reference frame. Throughout this paper, the term *location* will be used to refer to three variables: the robot's x and y coordinates and its heading direction θ. Although physically a robot always has a unique location ζ at any point in time, internally it only has a belief as to where it is located. BaLL describes this belief by a probability density over all locations ζ Є □, denoted by *bel(*ζ) where □ denotes the space of all locations. Occasionally we will distinguish the belief before taking a sensor snapshot, denoted by $bel_{\text{prior}}(\zeta)$, and the belief after incorporating sensor information, denoted by $bel_{\text{posterior}}(\zeta)$. The problem of localization is to approximate as closely as possible the "true" distribution of the robot location, which has a single peak at the robot's location and is zero elsewhere.

Each motion command (e.g., translation, rotation) changes the location of the robot. Expressed in probabilistic terms, the effect of a motion command $a \in A$, where A is the space of all motion commands, is described by a transition density:

$$P(\xi \mid \tilde{\xi}, a),$$

Which specifies the probability that the robot's location is ζ, given that it was previously at ξ and that it just executed action a. In practice it usually suffices to know a pessimistic approximation, which can easily be derived from the robot's kinematics/dynamics.

If the robot would *not* use its sensors, it would gradually lose information as to where it is

Due to slippage and drift (i.e., the entropy *bel(*ζ) would increase). Incorporating sensor readings counteracts this effect, since sensor measurements convey information about the robot's location.

5.3.5.1.2. Sensing

Let *S* denote the space of all sensor measurements (sensations) and let $s \in S$ denote a single sensation, where sensations depend on the location ζ of the robot. Let P(s | ζ).

Denote the probability that *s* is observed at location ζ. In practice, computing meaningful estimates of P(s | ζ) is difficult in most robotic applications. For example, if the robot's sensors include a camera, P(s | ζ) would be a highdimensional density capable of determining the probability of every possible camera image that could potentially be taken at any location ζ. Even if a full blown model of the environment is available, computing P(s | ζ) will be a complex, real time problem in computer graphics. Moreover, the current work does not assume that a model of the environment is given to the robot; hence, P(s | ζ) must be estimated from data.

To overcome this problem, it is common practice to extract (filter) a lowerdimensional feature vector from the sensor measurements. For example, landmark-based

approaches scan the sensor input for the presence or absence of landmarks, neglecting all other information contained therein. Model matching approaches extract partial models such as geometric maps from the sensor measurements, which are then compared to an existing model of the environment. Only the result of this comparison (typically a single value) is then considered further.

To formally model the extraction of features from sensor data, let us assume sensor data are projected into a smaller space *F*, and the robot is given a function

$$\sigma : S \longrightarrow F$$

Which maps sensations $s \in S$ into features $f \in F$. Borrowing terms from the signal processing literature, _ will be called a *filter* and the result of filtering a sensor reading $f=\sigma(s)$ will be called a *feature vector*. Instead of having to know P(s | ζ), it now suffices to know P(f | ζ), where P(f | ζ) relates the sensory features to different locations of the environment, for which reason it is often called a *map of the environment.*

The majority of localization approaches described in the literature assumes that the map is given. The probability P(f | ζ) can also be learned from examples. P(f | ζ), is often represented by a piecewise constant function (Buhmann et al. 1995; Burgard et al. 1996a; Burgard, et al., 1996b; Kaelbling, Cassandra, & Kurien, 1996; Koenig & Simmons, 1996; Moravec & Martin, 1994; Nourbakhsh, Powers, & Birchfield, 1995; Simmons & Koenig, 1995), or a parameterized density such as a Gaussian or a mixture of Gaussians (Gelb, 1974; Rencken, 1995; Smith& Cheeseman, 1985; Smith, Self, & Cheeseman, 1990). Below, in our experimental comparison, a *k*-nearest neighbor algorithm will be used to represent P(f|ζ).

In landmark-based localization, for example, σ filters out informationby recording only the presence and absence of individual landmarks, and P(f | ζ) models the likelihood of observing a landmark at the various locations ζ. P(f | ζ) can be estimated from data. The mathematically inclined reader may notice that the use of σ(s) instead of s is mathematically justified only if σ is a *sufficient statistic* (Vapnik, 1982) for estimating location-otherwise, all approaches that filter sensor data may yield suboptimal results (by ignoring important sensor information).

In practice, the suboptimality is tolerated, since P(f | ζ), or an approximate version of P(f|ζ), is usually much easier to obtain than P(s | ζ), and often is a good approximation to this probability.

5.3.5.1.3. Robot localization

For reasons of simplicity, let us assume that at any point in time *t*, the robot queries its sensors and then executes an action command that terminates at time *t+1*. In response to the sensor query, the robot receives a sensor reading $s^{(t)}$, from which it extracts a feature vector $f^{(t)}$. Let $f^{(1)}, f^{(2)}, \ldots = \sigma(s^{(1)}), \sigma(s^{(2)}), \ldots$ denote the sequence of feature vectors, and let $a^{(0)}, a^{(1)}, \ldots$ denote the sequence of actions. Furthermore, let $\zeta^{(0)}, \zeta^{(1)}, \ldots$denote the sequence of robot locations. Occasionally, locations will annotated by a * to distinguish them from variables used for integration.

Initially, at time *t=0*, the robot has a *prior belief* as to what its location might be; this prior belief is denoted $bel_{prior}(\zeta^{(0)})$ and reflects the robot's initial uncertainty. If the robot knows its initial location and the goal of localization is to compensate slippage and drift, $bel_{pri}(\zeta^{(0)})$ is a point-centered distribution that has a peak at the correct

location. The corresponding localization problem is called *position tracking*. Conversely, if the robot has no initial knowledge about its position, $bel_{prior}(\zeta^{(0)})$ is a uniform distribution. Here the corresponding localization problem is called *self localization*, *global localization*, or the "kidnapped robot problem" (Engelson, 1994), a task that is significantly more difficult than position tracking.
Sensor queries and actions change the robot's internal belief. Expressed probabilistically, the robot's belief after executing the *t-1*th action is:

$$Bel_{\text{prior}}(\xi^{(t)}) \;=\; P(\xi^{(t)}|f^{(1)}, a^{(1)}, f^{(2)}, a^{(2)}, \ldots, f^{(t-1)}, a^{(t-1)})$$

And after taking the *i*-th sensor measurement it is:

$$Bel_{\text{posterior}}(\xi^{(t)}) \;=\; P(\xi^{(t)}|f^{(1)}, a^{(1)}, f^{(2)}, a^{(2)}, \ldots, a^{(t-1)}, f^{(t)})$$

We will treat these two cases separately, starting with the second one.

5.3.5.1.3.1. Sensing

According to Bayes' rule,

$$\begin{aligned} Bel_{\text{posterior}}(\xi^{(t)}) \;&=\; P(\xi^{(t)}|f^{(1)}, \ldots, a^{(t-1)}, f^{(t)}) \\ &=\; \frac{P(f^{(t)}|\xi^{(t)}, f^{(1)}, \ldots, a^{(t-1)})\; P(\xi^{(t)}|f^{(1)}, \ldots, a^{(t-1)})}{P(f^{(t)}|f^{(1)}, \ldots, a^{(t-1)})}. \end{aligned}$$

The Markov assumption states that sensor readings are conditionally independent of previous sensor readings and actions given knowledge of the exact location:

$$P(s^{(t)}|\xi^{(t)}) \;=\; P(s^{(t)}|\xi^{(t)}, s^{(1)}, a^{(1)}, \ldots, a^{(t-1)})).$$

Since $f^{(t)} = \sigma(s^{(t)})$, it follows that

$$P(s^{(t)}|\xi^{(t)}) \;=\; P(f^{(t)}|\xi^{(t)}, f^{(1)}, a^{(1)}, \ldots, a^{(t-1)})).$$

It is important to notice that the Markov assumption does not specify the independence of different sensor readings if the robot's location is unknown; neither do it make assumptions on the extent to which $\zeta^{(t)}$ is known during localization. In mobile robot localization, the location is usually unknown-otherwise there would not be a localization problem-, and subsequent sensor readings and actions usually depend on each other. See Chung (1960), Howard (1960), Mine and Osaki (1970), and Pearl (1988) for more thorough treatments of conditional independence and Markov chains. The Markov assumption simplifies (8), which leads to the important formula (Moravec, 1988; Pearl, 1988):

$$\begin{aligned} Bel_{\text{posterior}}(\xi^{(t)}) \;&=\; \frac{P(f^{(t)}|\xi^{(t)})\; P(\xi^{(t)}|f^{(1)}, \ldots, a^{(t-1)})}{P(f^{(t)}|f^{(1)}, \ldots, a^{(t-1)})} \\ &=\; \frac{P(f^{(t)}|\xi^{(t)})\; Bel_{\text{prior}}(\xi^{(t)})}{P(f^{(t)}|f^{(1)}, \ldots, a^{(t-1)})}\;. \end{aligned}$$

The denominator on the right hand side of (11) is a normalizer which ensures that the belief $bel_{posterior}(\zeta^{(t)})$ integrates to 1. It is calculated as:

$$\begin{aligned} P(f^{(t)}|f^{(1)}, \ldots, a^{(t-1)}) \;&=\; \int_{\Xi} P(f^{(t)}|\xi^{(t)})\; P(\xi^{(t)}|f^{(1)}, \ldots, a^{(t-1)})\; d\xi^{(t)} \\ &=\; \int_{\Xi} P(f^{(t)}|\xi^{(t)})\; Bel_{\text{prior}}(\xi^{(t)})\; d\xi^{(t)}\;. \end{aligned}$$

To summarize, the posterior belief $bel_{posterior}(\zeta^{(t)})$ after observing the t-th feature vector $f^{(t)}$ is proportional to the prior belief $bel_{posterior}(\zeta^{(t)})$ multiplied by the likelihood $P(f^{(t)}|\zeta^{(t)})$ of observing f^{t} at $\zeta^{(t)}$.

5.3.5.1.3.2. Acting

Actions change the location of the robot and thus its belief. Recall that the belief after executing the t-th action is given by

$$Bel_{\text{prior}}(\xi^{(t+1)}) \;=\; P(\xi^{(t+1)}|f^{(1)},\ldots,f^{(t)},a^{(t)}) \;,$$

Table 1. The incremental localization algorithm.

1. Initialization: $Bel(\xi) \longleftarrow Bel_{\text{prior}}(\xi^{(0)})$
2. For each observed feature vector $f = \sigma(s)$ do:

$$Bel(\xi) \;\longleftarrow\; P(f|\xi)\; Bel(\xi) \tag{17}$$

$$Bel(\xi) \;\longleftarrow\; Bel(\xi)\;\left[\int_{\Xi} Bel(\tilde{\xi})\; d\tilde{\xi}\right]^{-1} \qquad \text{(normalization)} \tag{18}$$

3. For each action command a do:

$$Bel(\xi) \;\longleftarrow\; \int_{\Xi} P(\xi|\tilde{\xi},a)\; Bel(\tilde{\xi})\; d\tilde{\xi} \tag{19}$$

Which can be rewritten using the theorem of total probability as:

$$\int_{\Xi} P(\xi^{(t+1)}|\xi^{(t)},f^{(1)},\ldots,f^{(t)},a^{(t)})\; P(\xi^{(t)}|f^{(1)},\ldots,f^{(t)},a^{(t)})\; d\xi^{(t)}.$$

Since $\zeta^{(t)}$ does not depend on the action $a^{(t)}$ executed there, is equivalent to:

$$\int_{\Xi} P(\xi^{(t+1)}|\xi^{(t)},f^{(1)},\ldots,f^{(t)},a^{(t)})\; P(\xi^{(t)}|f^{(1)},\ldots,f^{(t)})\; d\xi^{(t)}.$$

By virtue of the Markov assumption, which if $\zeta^{(t)}$ is known renders conditional independence of $\zeta^{(t+1)}$ from $f^{(1)}$, $a^{(1)}$, ... , $\zeta^{(t)}$ (but not from $a^{(t)}$), $bel_{pri}(\zeta^{(t+1)})$ can be expressed as :

$$\int_{\Xi} P(\xi^{(t+1)}|\xi^{(t)},a^{(t)})\; P(\xi^{(t)}|f^{(1)},\ldots,f^{(t)})\; d\xi^{(t)}$$

$$\text{or}\quad \int_{\Xi} P(\xi^{(t+1)}|\xi^{(t)},a^{(t)})\; Bel_{\text{posterior}}(\xi^{(t)})\; d\xi^{(t)} \;.$$

5.3.5.2. The Bayesian localization error

This section and the following one present BaLL, a method for learning σ. The input to the BaLL algorithm is a set of sensor snapshots labeled by the location at which they were taken:

$$X = \{\langle s_k, \xi_k \rangle \mid k = 1, \ldots, K\} ,$$

Where K denotes the number of training examples. Localization is a specific form of state estimation. As it is common practice in the statistical literature on state estimation (Vapnik, 1982; Casella & Berger, 1990), the effectiveness of an estimator will be judged by measuring the expected deviation between estimated and true locations. BaLL learns σ by minimizing this deviation.

5.3.5.2.1. The posterior error $_{\text{posterior}}$

The key to learning σ is to minimize the localization error. To analyze this error, let us examine the update rule (17) in Table 1. This update rule transforms a prior belief to a refined, posterior belief, which is usually more accurate. Obviously, the posterior belief and thus the error depend on σ, which determines the information extracted from sensor data *s*.

Let ζ^* denote the *true* location of the robot (throughout the derivation, we will omit the time index to simplify the notation), and let $e(\zeta^*, \zeta)$ denote an error function for measuring the error between the true position ζ^* and an arbitrary other position ζ. The concrete nature of *e* is inessential to the basic algorithm; for example, *e* might be the Kullback-Leibler divergence or a metric distance.

The *Bayesian localization error at* ζ^*, denoted by $E(\zeta^*)$, is obtained by integrating the error *e* over all belief positions ζ , weighted by the likelihood $bel(\zeta)$ that the robot assigns to ζ, giving

$$E(\xi^*) = \int_{\Xi} e(\xi^*, \xi)\, Bel(\xi)\, d\xi .$$

If this error is computed prior to taking a sensor snapshot, that is, if $bel(\zeta) = bel_{prior}(\zeta)$, it is called the *prior Bayesian error at* ζ^* with respect to the next sensor reading and will be denoted E_{prior}. The prior localization error is a function of $bel_{pri}(\zeta)$.

We are now ready to derive the Bayesian error *after* taking a sensor snapshot. Recall that ζ^* denotes the true location of the robot. By definition, the robot will sense a feature vector *f* with probability $P(f \mid \zeta^*)$. In response, it will update its belief according to Equation (17). The *posterior Bayesian error at* ζ^*, which is the error the robot is expected to make at ζ^* after sensing, is obtained by applying the update rule (17) to the error (21), giving:

$$E_{\text{posterior}}(\xi^*) = \int_{\Xi} e(\xi^*, \xi)\, Bel_{\text{posterior}}(\xi)\, d\xi$$

$$= \int_{\Xi} e(\xi^*, \xi) \int_{F} \frac{P(f|\xi)\, Bel_{\text{prior}}(\xi)}{P(f)} P(f|\xi^*)\, df\, d\xi ,$$

Where E_{prior} posterior is averaged over all possible sensor feature vectors *f* weighted by their likelihood $P(f \mid \zeta^*)$. The normalizer $P(f)$ is computed just as in equations (12) or (18). Thus far, the posterior error E_{prior} corresponds to a single position ζ^* only. By averaging over all possible positions ζ^*, weighted by their likelihood of occurrence $P(\zeta^*)$, we obtainthe *average posterior error*

$$E_{\text{posterior}} = \int_{\Xi} E_{\text{posterior}}(\xi^*)\ P(\xi^*)\ d\xi^*$$
$$= \int_{\Xi}\int_{\Xi} e(\xi^*,\xi) \int_{F} \frac{P(f|\xi)\ Bel_{\text{prior}}(\xi)}{P(f)}\ P(f|\xi^*)\ P(\xi^*)\ df\ d\xi\ d\xi^*\ .$$

Since $f = \sigma(s)$, expression (23) can be rewritten as

$$E_{\text{posterior}} = \int_{\Xi}\int_{\Xi} e(\xi^*,\xi) \int_{S} \frac{P(\sigma(s)|\xi)\ Bel_{\text{prior}}(\xi)}{P(\sigma(s))}\ P(\sigma(s)|\xi^*)\ P(\xi^*)\ ds\ d\xi\ d\xi^*$$
$$\text{where}\ \ P(\sigma(s)) = \int_{\Xi} P(\sigma(s)|\tilde{\xi})\ Bel_{\text{prior}}(\tilde{\xi})\ d\tilde{\xi}\ .$$

The error E_{prior} is the exact localization error after sensing.

5.3.5.2.2. Approximating $E_{posterior}$

While $E_{posterior}$ measures the "true" Bayesian localization error, it cannot be computed in any but the most trivial situations (since solving the various integrals in (24) is usually mathematically impossible). However, $E_{posterior}$ can be approximated using the data. Recall that to learn σ, the robot is given a set of K examples:

$$X = \{\langle s_k, \xi_k\rangle \mid k = 1,\ldots,K\}\ ,$$

Where X consists of K sensor measurements s_k that are labeled by the location ζ_k at which they were taken. K is used to approximate $E_{posterior}$ with the expression:

$$\tilde{E}_{\text{posterior}} = \sum_{\langle\xi^*,s^*\rangle\in X}\ \sum_{\langle\xi,s\rangle\in X} e(\xi^*,\xi)\ \frac{P(\sigma(s)|\xi)\ Bel_{\text{prior}}(\xi)}{P(\sigma(s))} P(\sigma(s)|\xi^*)\ P(\xi^*),$$
$$\text{where}\ \ P(\sigma(s)) = \sum_{\langle\tilde{\xi},\tilde{s}\rangle\in X} P(\sigma(s)|\tilde{\xi})\ Bel_{\text{prior}}(\tilde{\xi})\ .$$

Leaving problems of small sample sizes aside, $E_{posterior}$ lets the robot compare different σ with each other: the smaller $E_{posterior}$, the better σ for the purpose of localization. This alone is an important result, as it lets one *compare* two filters to each other.

The error $E_{posterior}$ is a function of the prior uncertainty $bel_{pri}(\zeta)$as well. As a result, a specific σ that is optimal under one prior uncertainty can perform poorly under another. This observation matches our intuition: when the robot is globally uncertain, it is usually advantageous to consider different features than when it knows its location within a small margin of uncertainty.

5.3.5.3. The BaLL algorithm

BaLL learns the filter σ by minimizing $E_{posterior}$ through search in the space of filters σ, that is, by computing:

$$\sigma = \operatorname*{argmin}_{\hat{\sigma}\in\Sigma} \tilde{E}_{\text{posterior}}(\hat{\sigma})\ ,$$

Where □is a class of functions from which σ is chosen. This section presents a specific search space □, for which it derives a gradient descent algorithm.

5.3.5.3.1. Neural network filters

BaLL realizes σ by a collection of n backpropagation style *feed*-forward artificial neural networks (Rumelhart, Hinton, & Williams, 1986). Each network, denoted by g_i

with *i=1,...,n* , maps the sensor data *s* to a feature value in (0,1). More formally, we have

$$\sigma = (g_1, g_2, \ldots, g_n) \ ,$$
$$\text{where for all } i = 1, \ldots, n,$$
$$g_i : S \longrightarrow (0, 1)$$

is realized by an artificial neural network. The *i*-th network corresponds to the *i*-th feature, where *n* is the dimension of the feature vector *f*.
Neural networks can approximate a large class of functions (Hornik, Stinchcombe, & White, 1989). Thus, there are many features that a neural network can potentially extract. To the extent that neural networks are capable of recognizing landmarks, our approach lets a robot automatically select its own and learn routines for their recognition.

5.3.5.3.2. Stochastic filters

At first glance, it might seem appropriate to define $f = (g_1(s), g_2(s), \ldots, g_n(s))$, making the feature vector *f* be the concatenated n-dimensional output of the *n* neural networks.
Unfortunately, such a definition would imply $F = (0,1)^n$, which contains an infinite number of feature vectors *f* (since neural networks produce real-valued outputs). If the sensor readings are noisy and distributed continuously, as is the case for most sensors used in today's robots, the chance is zero that two different sensations taken at the same location will generate the same feature vector *f*. In other words, if define $f = (g_1(s), g_2(s), \ldots, g_n(s))$, *F* would be too large for the robot to ever recognize a previous location-a problem that specifically occurs when using real-valuedfunction approximators as feature detectors.
Fortunately, there exists an alternative representation that has several nice properties. In the BaLL algorithm $F = \{0,1\}^n$ and $|F| = 2^n$ (which is finite). Each neural network is interpreted as a *stochastic* feature extractor, which generates the value $f_i = 1$ with probability $g_i(s)$ and the value $f_i = 0$ with probability $1 - g_i(s)$, giving:

$$P(f_i = 1|s) \ = \ g_i(s)$$
$$P(f_i = 0|s) \ = \ 1 - g_i(s) \ .$$

We assume that the joint probability $P(f \mid s)$ is given by the product of the marginal probabilities $P(f_i \mid s)$:

$$P(f|s) \ = \ \prod_{i=1}^{n} P(f_i|s) \ .$$

The stochastic setting lets σ express confidence in its result by assigning probabilities to the different $f \in F$ - a generally desirable property for a filter.
The stochastic representation has another advantage, which is important for the efficiency of the learning algorithm. As we show below, $E_{posterior}$ is differentiable in the output of the function approximator and hence in the weights and biases of the neural networks. Differentiability is a necessary property for training neural networks with gradient descent.

5.3.5.3.3. The neural network learning algorithm

The new, stochastic interpretation of σ requires that E_{post} and its approximation $E_{posterior}$ be modified to reflect the fact that σ generates a probability distribution over F instead of a single $f \in F$. Following the theorem of total probability and using (23) as a starting point, E_{post} is given by:

$$E_{\text{posterior}} = \int_{\Xi}\int_{\Xi} e(\xi^*,\xi) \sum_{f=(0,\ldots,0)}^{f=(1,\ldots,1)} \frac{\int_S P(f|s)P(s|\xi)\ Bel_{\text{prior}}(\xi)\ ds}{P(f)} \int_S P(f|s)P(s|\xi^*)\ P(\xi^*)\ ds\ d\xi\ d\xi^*, \quad (34)$$

$$\text{where } P(f) = \int_{\Xi}\int_S P(f|s)\ P(s|\tilde{\xi})\ Bel_{\text{prior}}(\tilde{\xi})\ ds\ d\tilde{\xi}. \quad (35)$$

The approximation of this term is governed by

$$\tilde{E}_{\text{posterior}} = \sum_{\langle\xi^*,s^*\rangle\in X} \sum_{\langle\xi,s\rangle\in X} e(\xi^*,\xi) \sum_{f=(0,\ldots,0)}^{f=(1,\ldots,1)} \frac{P(f|s)\ Bel_{\text{prior}}(\xi)}{P(f)} P(f|s^*)\ P(\xi^*)$$

$$= \sum_{\langle\xi^*,s^*\rangle\in X} \sum_{\langle\xi,s\rangle\in X} e(\xi^*,\xi)\ P(\xi^*)\ Bel_{\text{prior}}(\xi) \sum_{f=(0,\ldots,0)}^{f=(1,\ldots,1)} \frac{P(f|s)}{P(f)} P(f|s^*), \quad (36)$$

$$\text{where } P(f) = \sum_{\langle\tilde{\xi},\tilde{s}\rangle\in X} P(f|\tilde{s})\ Bel_{\text{prior}}(\tilde{\xi}). \quad (37)$$

The mathematically inclined reader should notice that (24) and (27) are special cases of (34) and (36). They are equivalent if one assumes that *P(f | s)* is deterministic, that is, if
P(f | s) is centered on a single *f* for each *s*.
Armed with an appropriate definition of $E_{posterior}$, we are now ready to derive the gradient descent learning algorithm for training the neural network feature recognizers to minimize $E_{posterior}$. This is done by iteratively adjusting the *weights* and *biases* of the *i*-th neural network, denoted by $w_{i\mu\nu}$, in the direction of the negative gradients of $E_{posterior}$:

$$w_{i\mu\nu} \longleftarrow w_{i\mu\nu} - \eta \frac{\partial \tilde{E}_{\text{posterior}}}{\partial w_{i\mu\nu}}.$$

Here $\eta > 0$ is a learning rate, which is commonly used in gradient descent to control the magnitude of the updates. Computing the gradient in the right hand side of (38) is a technical matter, as both $E_{posterior}$ and neural networks are differentiable:

$$\frac{\partial \tilde{E}_{\text{posterior}}}{\partial w_{i\mu\nu}} = \sum_{\langle\bar{\xi},\bar{s}\rangle\in X} \frac{\partial \tilde{E}_{\text{posterior}}}{\partial g_i(\bar{s})} \frac{\partial g_i(\bar{s})}{\partial w_{i\mu\nu}}.$$

The second gradient on the right hand side of (39) is the regular outputweight gradient used in the backpropagation algorithm, whose derivation we omit (see Hertz, Krogh, & Palmer, 1991; Rumelhart, Hinton, & Williams, 1986; Wasserman, 1989). The first gradient in (39) can be computed as:

$$\frac{\partial \tilde{E}_{\text{posterior}}}{\partial g_i(\bar{s})} \overset{(36)}{=} \sum_{\langle \xi^*, s^* \rangle \in X} \sum_{\langle \xi, s \rangle \in X} e(\xi^*, \xi)\ P(\xi^*)\ Bel_{\text{prior}}(\xi) \sum_{f_1=0}^{1} \sum_{f_2=0}^{1} \cdots \sum_{f_n=0}^{1} \frac{\partial}{\partial g_i(\bar{s})} \left[\left(\prod_{i=1}^{n} P(f_i|s^*)\ P(f_i|s) \right)\ P(f)^{-1} \right] \tag{40}$$

$$= \sum_{\langle \xi^*, s^* \rangle \in X} \sum_{\langle \xi, s \rangle \in X} e(\xi^*, \xi)\ P(\xi^*)\ Bel_{\text{prior}}(\xi) \sum_{f_1=0}^{1} \sum_{f_2=0}^{1} \cdots \sum_{f_n=0}^{1} \prod_{j \neq i} P(f_j|s^*)\ P(f_j|s) \cdot \left[\frac{\delta_{\xi^*,\bar{\xi}} P(f_i|s) + \delta_{\xi,\bar{\xi}} P(f_i|s^*)}{\sum_{\langle \tilde{\xi}, \tilde{s} \rangle \in X} \prod_{j=1}^{n} P(f_j|\tilde{s})} - \frac{P(f_i|s^*) P(f_i|s) \prod_{j \neq i} P(f_j|\bar{s}) Bel_{\text{prior}}(\bar{\xi})}{\left(\sum_{\langle \tilde{\xi}, \tilde{s} \rangle \in X} \prod_{j=1}^{n} P(f_j|\tilde{s}) \right)^2} \right] (2\delta_{f_i,1} - 1)\ .$$

Here $\delta_{x,y}$ denotes the Kronecker symbol, which is 1 if $x=y$ and 0 if $x \neq y$. $P(f_j \mid s^*)$ is computed according to Equation (32).

Table 2 describes the BaLL algorithm and summarizes the main formulas derived in this and the previous section. BaLL's input is the data set X and a specific prior belief $bel_{pri}(\zeta)$. Below, we will train networks for different prior beliefs characterized by different entropies (i.e., degrees of uncertainty). The gradient descent update is repeated until one reaches a termination criterion (e.g., early stopping using a crossvalidation set or pseudoconvergence of $E_{posterior}$), as in regular backpropagation (Hertz, Krogh, & Palmer, 1991).3

BaLL differs from conventional backpropagation (supervised learning) in that no target values are generated for the outputs of the neural networks. Instead, the quantity of interest, $E_{posterior}$, is minimized directly. The output characteristics of the individual networks and, hence, the features they extract, emerge as a side effect of minimizing E_{post}.

The output of the BaLL algorithm is a set of filters specified by a set of weights and biases for the different networks. As noted above, $E_{posterior}$ and the resulting filter σ depend on the uncertainty $bel_{perior}(\zeta)$. Below, when presenting experimental results, we will show that, in cases in which the uncertainty is small, quite different features are extracted than when the uncertainty is large. However, although the networks must be *trained* for a particular $bel_{perior}(\zeta)$, they can be *used* to estimate the location for arbitrary uncertainties $bel_{pri}(\zeta)$, but with degraded performance. It is therefore helpful, but not necessary, to train different networks for different prior uncertainties.

Table 2. BaLL, the algorithm for learning neural network filters σ.

Input: Data set $X = \{\langle s_k, \xi_k \rangle \mid k = 1, \ldots, K\}$, prior belief $Bel_{\text{prior}}(\xi)$.

Output: Optimized parameters (weights and biases) $w_{i\mu\nu}$ for the n networks $g_1, \ldots, g_n$.

Algorithm:

1. Initialize the parameters $w_{i\mu\nu}$ of every network with small random values.
2. Iterate until convergence criterion is fulfilled:

   2.1 For all $\langle \xi, s \rangle \in X$, compute the conditional probabilities

$$P(f_i|s) \;=\; \begin{cases} g_i(s) & \text{if } f_i = 1 \\ 1 - g_i(s) & \text{if } f_i = 0 \end{cases} \tag{41}$$

   where $g_i(s)$ is the output of the i-th network for input s (cf. (32)).

   2.2 Compute the error $\tilde{E}_{\text{posterior}}$ (cf. (36))

$$\tilde{E}_{\text{posterior}} \;=\; \sum_{\langle \xi^*, s^* \rangle \in X} \sum_{\langle \xi, s \rangle \in X} e(\xi^*, \xi)\; P(\xi^*)\; Bel_{\text{prior}}(\xi) \cdot \sum_{f_1=0}^{1} \sum_{f_2=0}^{1} \cdots \sum_{f_n=0}^{1} \left(\prod_{i=1}^{n} P(f_i|s^*)\; P(f_i|s) \right) \left[\sum_{\langle \tilde{\xi}, \tilde{s} \rangle \in X} \left(\prod_{i=1}^{n} P(f_i|\tilde{s}) \right)\; Bel_{\text{prior}}(\tilde{\xi}) \right]^{-1} \tag{42}$$

   2.3 For all network parameters $w_{i,\mu,\nu}$, compute

$$\frac{\partial \tilde{E}_{\text{posterior}}}{\partial w_{i\mu\nu}} \;=\; \sum_{\langle \bar{\xi}, \bar{s} \rangle \in X} \frac{\partial g_i(\bar{s})}{\partial w_{i\mu\nu}} \sum_{\langle \xi^*, s^* \rangle \in X} \sum_{\langle \xi, s \rangle \in X} e(\xi^*, \xi)\; P(\xi^*)\; Bel_{\text{prior}}(\xi) \cdot \sum_{f_1=0}^{1} \sum_{f_2=0}^{1} \cdots \sum_{f_n=0}^{1} \prod_{j \neq i} P(f_j|s^*)\; P(f_j|s)\; (2\delta_{f_i,1} - 1) \tag{43}$$

$$\cdot \left[\frac{\delta_{\xi^*, \bar{\xi}}\, P(f_i|s) + \delta_{\xi, \bar{\xi}}\, P(f_i|s^*)}{\sum_{\langle \tilde{\xi}, \tilde{s} \rangle \in X} \prod_{j=1}^{n} P(f_j|\tilde{s})} - \frac{P(f_i|s^*)\; P(f_i|s) \prod_{j \neq i} P(f_j|\bar{s})\; Bel_{\text{prior}}(\bar{\xi})}{\left(\sum_{\langle \tilde{\xi}, \tilde{s} \rangle \in X} \prod_{j=1}^{n} P(f_j|\tilde{s}) \right)^2} \right].$$

   The gradients $\dfrac{\partial g_i(\bar{s})}{\partial w_{i\mu\nu}}$ are obtained with backpropagation (cf. (39) and (40)).

   2.4 For all network parameters $w_{i,\mu,\nu}$, update (cf. (38))

$$w_{i\mu\nu} \;\longleftarrow\; w_{i\mu\nu} - \eta\, \frac{\partial \tilde{E}_{\text{posterior}}}{\partial w_{i\mu\nu}}. \tag{44}$$

5.3.5.3.4. Algorithmic complexity

The complexity of the learning and the performance methods must be analyzed separately. The localization algorithm described in Table 1 must be executed in real time, while the robot is in operation, whereas the learning algorithm described in Table 2 can be run offline. Our primary concern in the analysis is time complexity.

5.3.5.3.4.1. Localization

The complexity of probabilistic localization (Table 1) depends on the representation of $P(f|\zeta)$ and $bel(\zeta)$. In the worst case, processing a single sensor reading requires $O(Kn+nW)$ time, where K is the training set size, n is the number of networks and W is the number of weights and biases in each neural network. Processing an action requires $O(K^2n)$ time. Various researchers have implemented versions of the probabilistic localization algorithm that work in real time (Burgard et al., 1996a; Burgard, Fox, & Thrun, 1997; Kaelbling, Cassandra, & Kurien, 1996; Koenig & Simmons, 1996; Nourbakhsh, Powers, & Birchfield, 1995; Simmons & Koenig, 1995; Thrun et al., 1996; Thrun, 1996). Given the relatively small computational overhead of the existing implementations, scaling to larger environments is not problematic.

5.3.5.3.4.2. Learning

BaLL requires $O(N2^nK^3 + NKnW)$ time, where n , K , and W are the same as above, and where N is the number of gradient descent iterations. If the number of training patterns is greater than both the number of inputs and the number of hidden units in each network, which is a reasonable assumption since otherwise the number of free parameters exceeds the number of training patterns by a huge margin, then $O(N2^nK^3)$ dominates $O(NKnW)$.Thus, under normal conditions, the training the networks requires $O(N2^nK^3)$ time. The constant factor is small (cf. Table 2). Most existing localization algorithms use only one or two features (e.g., one or two landmarks), indicating that even small values for n work well in practice.

There are several ways to reduce the complexity of learning:

1. Instead of training all networks in parallel, they can also be trained one after another, similar to the way units are trained one after another in the cascade correlation algorithm (Fahlman & Lebiere, 1989). Sequential training would reduce worstcase exponential to linear complexity, since networks are trained one after another, which requires $O(Nnk^3)$ time.
2. Compact representations for $P(f|\zeta)$ and $bel(\zeta)$ can reduce the complexity significantly. For example, in Burgard et al. (1996a), Koenig and Simmons (1996), and Simmons and Koenig (1995), the number of grid cells used to represent $P(f|\zeta)$ and $bel(\zeta)$ is independent of the training set size. Using their representations, our learning algorithm would scale quadratically in the size of the environment and linearly in the size of the training set. In addition, coarsegrained representations such as the one reported by Koenig and Simmons (1996) and Simmons and Koenig (1995) can reduce the constant factor even further.
3. The learning algorithm in Table 2 interleaves one computation of $E_{posterior}$ and its derivatives with one update of the weights and biases. Since the bulk of processing time is spent computing $E_{posterior}$ and its derivatives, the overall complexity can be reduced by modifying the training algorithm so that multiple updates of the networks'

parameters are interleaved with a single computation of $E_{posterior}$ and its derivatives. The necessary steps include:

i. The network outputs $g_i(s)$ are computed for each training example $(s, z) \in X$.
ii. The gradients of $E_{posterior}$ with respect to the network outputs $g_i(s)$ are computed.
iii. For each training example $(s, z) \in X$, "pseudopatterns" are generated using the current network output in conjunction with the corresponding gradients, giving:

$$\left\langle \; s \; , \;\; g_i(s) - \frac{\tilde{E}_{\text{posterior}}}{g_i(s)} \right\rangle \; .$$

iv. These patterns are fitted using multiple epochs of regular backpropagation. This algorithm approximates gradient descent, but it reduces the complexity by a constant factor.

In addition, modifications such as online learning, stochastic gradient descent, or higherorder methods such as momentum or conjugate gradient methods (Hertz, Krogh, & Palmer, 1991) yield further speedup. Little is currently known about principal complexity bounds that would apply here.
As noted above, learning σ can be done offline and is only done once. With the modifications proposed here, the complexity of training is low order polynomial (mostly linear) in *K, n, N* and *W*. In the light of the modifications discussed here, scaling up our approach to larger environments, larger training sets and more neural networks does not appear to be problematic.

5.4. Output Map

As you know there are two types of maps:

i) Absolute metric map
ii) Topological map

Because of advantages of topological map we use it to modify the environment. In this type of map we partition the environment and represent these partitions and the connections between them as a topological map [Tomatis et al., 2002; Bosse et al., 2003; Yeap and Jefferies, 1999; Kuipers, 2000]. Whilst the first approach explicitly represents the environment in absolute metric terms, the second often combines both metric and topological information.
At this time we can generate a good 2D map as shown in figure x but we are developing our system for generating efficient 3D map. This is done by 3D landmarking.
Figure x shows the generated map by Arian III in our Lab building. This map is colored and has constructed by merging partitions of the topological map.

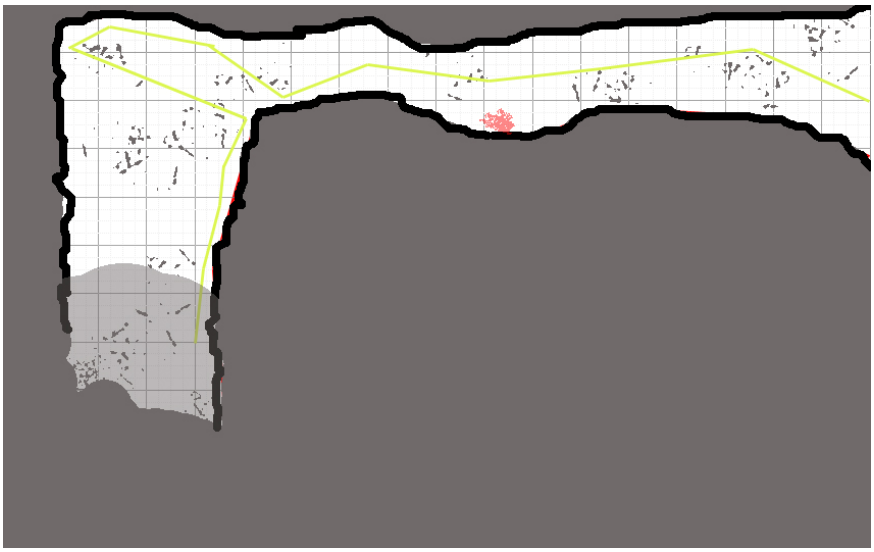


Fig. 34 Generated map by Arian III in Lab building.

6. Sensors for Navigation and Localization

Various navigation systems are preparing necessary equipments for suitable robot mobility. Although this equipments are considering by an intelligent operator but each of them are able to the extent that could handle the robots automatic control system without the leader. The specifications such as distinguishing the impracticable obstacles, finding the best way, robot navigation and amending the routes are also the robot navigation tools capabilities.

6.1 Laser Range Scanner

Several possibilities were considered for a primary distance measuring device for the around of the robots. Laser is the best choice for this purpose because laser emitted a beam and there is no ping from other devices. We decided to use The Hokuyo laser range finder Model URG-04LX that has good resolution in Distance and Angel Also the price of this scanner is economical and its size is very suitable for robots.

Fig. 35 Laser line scanner

The outputs of this sensor may be very useful for operator between the competitions for avoiding of smash to partitions.
The output of laser scanner can not be sufficient for map generating rather this information must be complete with position of robot and output of many orientation sensor until we have a map of environment that robot is placed there.
The characteristics of this scanner are mentioned below:

Specifications

Products	Scanning Laser Range Finder
Model No.	URG-04LX
Light source	Semiconductor laser λ =785nm, Laser safety class 1(IEC60825-1)
Power source	5VDC, ±5%
Current consumption	500mA or less(rush current 800mA)
Detectable distance and objects	20mm to 4000mm, white Kent sheet 70mm x 70mm*
Accuracy	(Official 20 to 1000mm : ±10mm, 1000 to 4000mm : ±1% of measurement) (White Kent sheet 70mm x 70mm)
Resolution	1mm
Scanning angle	240°
Angle resolution	Approx. 0.36° (360°/1024 steps)
Scanning time	100msec/scan
Interface	RS-232C(19.2k, 57.6k, 115.2k, 500k, 750kbps), USB : Ver.2.0 FS mode(12Mbps)
Ambient temperature/ humidity	-10 to +50°C, 85%RH or less(without dew and frost)
Protective structure	Optics : IP64, Case : IP40
Weight	Approx.160g
Material	Polycarbonate

And there is the sample of scanning from the company catalog:

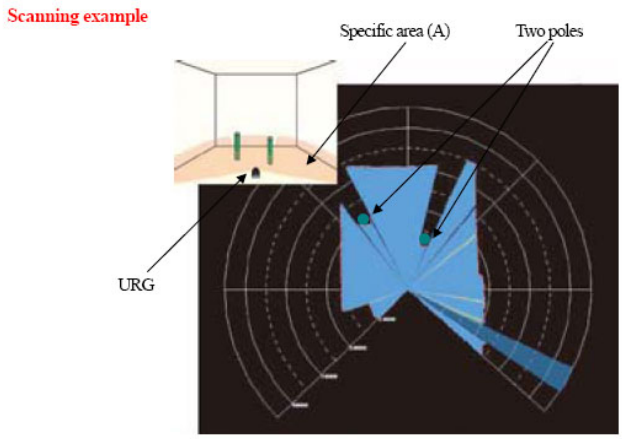


Fig. 36 Scanning Sample

6.2 IR ranger

The Sharp GP2D02 is a sensitive compact distance measuring sensor. It required two lines from a microcontroller in order to be controlled. One line provides the signal to begin a measurement and also is used to provide a clock signal when transmitting the distance measure and the other line is used to transmit the measurements back to the microcontroller.

The GP2D02 is a self contained device which emits an IR pulse and determines the distance of a nearby object using triangulation. It is able to measure distances up to 80 cm and at that range has a beam width of only 10 cm. I mounted this sensor on servo motor at the front of the robot. The servo sweeps the sensor through a 50 degree pattern. The servo is discussed in more detail later. The sensor is digitally controlled with its Vin line. The Vin line is pulsed low to tell the sensor to begin a measurement. The sensor will output a high on the Vout line when it is ready to transmit. The Vin line is then pulsed, and the sensor data is clocked in on the Vout line. This is illustrated in the figure below.

Figure 1 - GP2D02 Timing

GP2D02 Measurements

The distance measurement is a 8 bit number. It is not linear, as can be seen in Figure 2 below. The distance measurement can be linearized using the following formula proposed by Sean H. Breheny.

Linearized data = 1.9/(tan(reading*25/1000))

The constants in the formula above were established by Mr. Breheny through experimental means. Using my measured data from 3 GP2D02 sensors, I was unable to improve the linearization calculations by manipulating the constants, and the formula was implemented as is. I later discovered that reasonable linearization can be achieved by simply inverting the reading received (1/reading) and then multiplying it by some constant. This eliminates the tan term, which is not a pretty thing in a 8 bit

microcontroller to implement. <grin> Actually, I ended up not linearizing the data at all in the robot. I designed the robot around a behavioural model (ie- instincts rather than brains) and it was just as easy to use the real sensor readings. Linearizing the readings would be very useful if you were trying to create a map from what the robot was seeing. I will be looking deeper into this in a future design.

As can be seen from the graphs above, the sensors have a lower limit. At around 7cm, the value read peaks and then begins to fall again. In other words, an object closer then 7 cm will appear to be further away. If we do not 'see' the object before it enters this zone, this will become a large problem with this particular sensor. This is another good reason for the robots to move slowly. This is also an excellent reason for having bumpers on a robot.

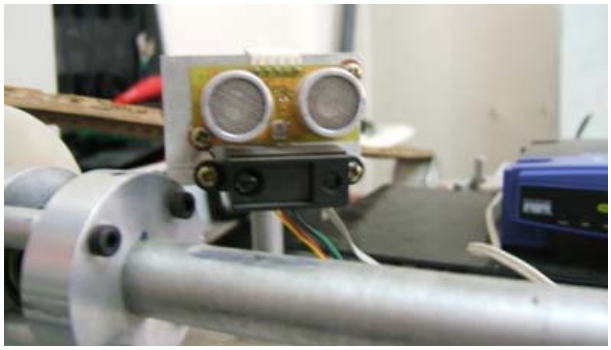


Fig. 37 IR Ranger and Ultrasonic sensors

6.3 Ultrasonic Rangers

This Devantech high performance ultrasonic range finder is compact and measures an amazingly wide range from 3cm to 6m. The SRF08 interfaces to your microcontroller via the industry standard IIC Bus.

This ranger is perfect for robots, or any other projects requiring accurate ranging information.

There is some of the Specification for SFR08:

Specifications

Beam Pattern	see graph
Voltage	5v
Current	15mA Typ. 3mA Standby
Frequency	40KHz
Maximum Range	6 m
Minimum Range	3 cm
Max Analogue Gain	Variable to 1025 in 32 steps
Connection	Standard IIC Bus
Light Sensor	Front facing light sensor
Timing	Fully timed echo, freeing host computer of task
Echo	Multiple echo - keeps looking after first echo
Units	Range reported n uS, mm or inches
Weight	0.4 oz.
Size	43mm w x 20mm d x 17mm h

Beam Pattern

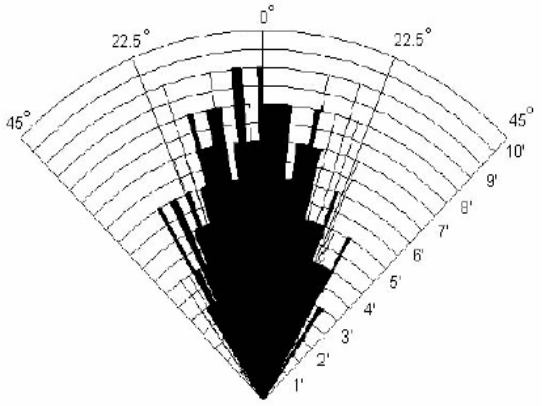


Fig. 38 Ultrasonic ranging sample

6.4 Electrical Compass

We used of The CMPS03 Magnetic Compass has been specifically designed for use in robots as an aid to navigation. The compass uses the Philips KMZ51 magnetic field sensor, which is sensitive enough to detect the Earths magnetic field. The output from two of them mounted at right angles to each other is used to compute the direction of the horizontal component of the Earths magnetic field. The bearing may be retrieved from the module in one of two ways; either using its PWM signal or its I2C interface.

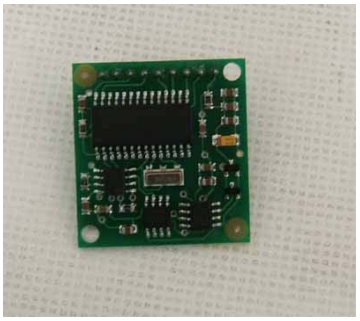


Fig. 39 Electrical Compass (CMD03)

Table.4

Voltage	5V
Current	20mA typical
Resolution	0.1 Degree
Accuracy	3-4 degrees approximately after calibration
Output 1	Timing Pulse (1ms to 37ms in 0.1ms increments)
Output 2	I2C Interface, 0-255 and 0-3599 SCL speed up to 1MHz
Dimensions	32mm x 35mm

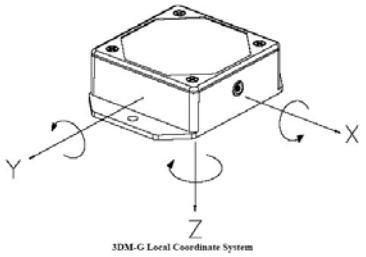
6.5 Gyro Enhanced Orientation Sensor

We have one of the best orientation sensor that buy from Microstrain Company called 3DM-G Gyro Enhanced Orientation Sensor.

The measurements output by the 3DM-G give you the orientation of the 3DM-G's local Coordinate system with respect to the Earth's coordinate system. If you orient the 3DMG Such that its Z-axis is pointing down through the center of the Earth, its X-axis is Pointing North and its Y-axis is pointing east; you have aligned the 3DM-G with Earth's Coordinate system. At this orientation the 3DM-G will be outputting the so-called

Fig. 40 3DM Sensor

'Identity matrix' which means the same as saying zero pitch, zero roll and zero yaw. If You turn it from there; you'll start getting non-zero pitch, roll and/or yaw.
Inside the 3DM-G The 3DM-G incorporates:
• 3 accelerometer sensors to measure Earth's gravity;
• 3 magnetometer sensors to measure magnetic fields;
• 3 rate gyroscope sensors to measure the rate of rotation about their sensitive axis;
• A temperature sensor;
• Signal conditioning amplifiers to condition the raw output of the sensors;
• A signal multiplexer to route the sensors' signals to the A/D converter;
• A 12-bit A/D converter that converts the conditioned output of the sensors into the digital domain;
• A microprocessor that carries out the processing algorithm;
• Non-volatile EEPROM to store calibration, filter and other parameters;
• And a data communications port.



3DM-G Local Coordinate System

6.6 GPS (Global Positioning System)
The Lassen iQ GPS receiver is a full featured, ultra low power receiver
On a miniature form factor, suitable for a variety of mobile, embedded applications. The Lassen iQ GPS receiver incorporates Trimble's first GPSTM architecture in the form of two ASICS: Colossus RF down Converter and IO-C33 base band chip.
The IO-C33 integrates Trimble's IO digital signal processor with the Epson C33 RISC processor, real-time clock, UART, and 1Mbit memory. Together with the colossus RF, this implementation of first GPS technology makes possible one of the smallest (26 mm x 26 mm x 6mm) and lowest power (less than 89 mW) GPS Modules available.
The Lassen iQ GPS receiver outputs a complete position, velocity, and Time (PVT) solution in the NMEA Version 3.0 ASCII protocol, the Trimble ASCII Interface Protocol (TAIP), and the Trimble TSIP Binary protocol. A Pulse-Per-Second signal is available for very accurate timing applications.

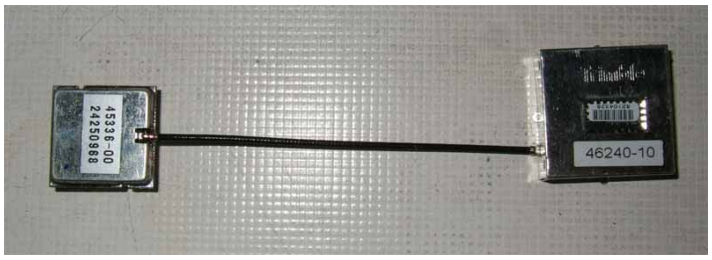


Fig. 41 GPS Receiver

The interface motherboard includes a 9 to 32 VDC switching power supply which provides regulated +3.3 VDC power to the receiver, and contains circuitry which provides two RS-232 interface ports. A 3.6V lithium backup battery enables quick hot starts. The TTL level PPS is brought directly out to Pin 9 of the Port 2 DB9 connector on the front of the interface unit.

6.7 Panoramic Camera

We have one Camera from SONY Technology with Full-Circle Lens Overview of SONY Company that gives us better view of surrounds.
The Output of this Camera Captured with Industrial Main system on Robot and with analyses in special FPGA with Image Processing help us to map generating.
This camera has the best Picture for 360 views and we prefer to use of this camera rather than a CCD camera patched with omni mirror.
one of the problem of omni mirrors is focus of center of mirror with lens of camera that problem is solved in this Camera.

Fig. 42 Panoramic camera

Fig. 43 Panoramic view

7. Sensors for Victim Identification

It is important collections of the feedbacks which are culminate in to the victim findings, receiving proper Information from the victim recognizer sensors on the robot. Attending to the real disaster site conditions and the simulated environment in the competition so it seems to be necessary to put minimum 3 kinds of these sensors on the robot. The rescue robot team has considered all kinds of the victim finder sensors in Order to receive the information from the environments feedbacks and have used the below tools and instruments, Which are the most important criterion according to the particulars such as the accuracy, speed, connecting to the computer system abilities.
We for locate of Victims have used of many sensors in our robot until we have different of solution for victim identification this sensors are:

7.1 Microphone and Speaker

In the rescuing conditions the voice emits can occur that can be received by the sensitive microphones. in addition to use the suitable tools for receiving the voice ,the automatic recognizing victim's voice system , is one of the team's activities in order to implementing the automatic specifying voice .
To detect the victims voice in the disaster sites, first of all we will consider the different methods of specifying the voice activities and then by selecting the best and useful methods we have started designing a proper system in fact distinguishing the voice activity is the same as specifying the human voices of silence which is frequently in the connection industries, the speech specifying and coding the speech and etc. is being used.
But unfortunately in spite of the various methods in this field most of the voice activities specifying styles in the noise environments where the ratio of voice to noise amount is low are including many problems.

Because of this using those methods in robot is based on their Improvement and optimization.
With this aim and in order to simulate the rescue competition acoustic environment among three human voice samples, the voice which is contaminated with noise and a noise sample have been used by order from these data bases: TIMIT, Spear, NOISEX.
We used of one sensitive microphone with Zoom capability to hear the sound of victims. This sound transmits to out of robot for hearing and analyses.
And We Have a Speaker for Alarm or Advise victim in Real Condition.

7.2 Thermal IR Camera

One of the solutions in victim identifying is detect of temperature of victims and in darkness this camera have better picture than CCD cameras.
Therefore we consider one of the best of IR Thermal camera in front of Robot.

Fig. 44 Thermal camera

This Camera has many functions for Example:

1) Zoom 3x
2) Digital Output
3) Analog output
4) Manual focus
5) Color Pattern

There are pictures of output of our IR camera with several patterns and normal picture:

Fig. 45 Thermal image

7.3 IR Thermometer

We have an IR thermometer with visible goal point for discover temperature of the victims. We use an Industrial Non-Contact Infrared Thermometer of Omega company with analog output.

Fig. 46 Temperature sensor

Accuracy:
±1% of reading @ 25°C ambient or 1.7°C (3°F), whichever is greater
Repeatability:
±1% rdg ±1 digit
Spectral Response:
8 to 14 microns
Emissivity Range:
0.10 to 1.00
Field of View (FOV):
See diagrams
Display:
Backlit LCD
Transmitter Outputs:
1 mV/degree, 0 to 5 Vdc or 4 to 20 mA
Power:
7 to 24 Vdc @ 80 mA
Environmental Ratings:
NEMA 4 water-tight and dust-tight for sensing head and electronics enclosure
Ambient Operating Range:
Sensing head 0 to 50°C

7.4 Co2 Sensor

One of the solutions of victim identification is find the source of co2 in environment. Since we decided to buy a sensitive co2 sensor of VALTRONICS Company with NDIR technology and digital signal processing and Temperature.

Fig. 47 CO2 Sensor Board

This sensor has digital output with SPI (Serial peripheral Interface) so have high speed responsibility that can very helpful to find of victim that are near but are spy and specify of sign of alive.

Table. 5

Model 2015SPI-1 Specifications:	
Method:	NDIR with Digital Signal processing and temperature compensation
Gas:	**Carbon Dioxide** (CO_2)
Range:	**0-2**% CO_2 (Full scale is user selectable anywhere from **0.2 to 2.0**%)
Input Power:	**+12 VDC** (@ 0.250 amp max., 0.135 amp ave., 16.0 volts max, 8.0 volts min)
Accuracy:	If calibrated at 0.5% CO_2 using 0.5±0.01% CO_2 gas, the accuracy is
	0 to 0.5±0.02% CO_2 and 5% of reading from 0.55 to 2% CO_2.
	16 bit analog to digital converter: Delta-Sigma Conversion Method, 12 bit DAC for 1V
Resolution / Repeatability :	**±0.002**% CO_2 (challenge with same gas sample multiple times & assure zero)
Stability:	Short term < 0.002% CO_2 in 20 sec., Long term: 0.5±0.1 or 1±0.2% CO_2 per year
Output Signal:	Digital **SPI** and linear **0** to **1** volt **output** signal See **Notes A59** and **A62**
Optional **RS-232** Serial Interface:	PCB for com. with any PC - see App. Note **A66** Type"**VTI**" to UNLOCK
LED Indicators:	IR Source ON / OFF Indicator, Power ON indicator, Cal Switch Indicators.
Input Signal:	Digital **SPI** input for calibration and diagnostic modes. See page 3
Calibration Switches:	SW1 (Zero), SW2 (Span Target), SW3 (Span), SW4 (Range adj), remote via J3
Operating Temperature Range:	0 to 50°C (32° to 122°F) see **Application Note A12**
Ambient Relative Humidity:	0 to 95% RH non-condensing: see **Application Note A30**
Storage Temperature range:	-40 to +70°C (-40 to +158°F)
Weight:	Less than 0.25 pound (<0.11 kilogram)
External Dimensions (PCB Card):	**4.9**" x **2.9**" x **3**" see page 2 for mounting

7.5 Camera

Having sufficient video images with a good quality is one of the most important rescue robots succeed factors seeking an injured person. For this reason we have used 3 cameras with a good view angle till the Ability of cameras can help the conductor of robot. Because the robot height is less than the sight level of standing a human, controlling of the robot from the cameras is very difficult and sensitive that more over it needs experience and practice in control.

The Front camera is a CCD Panasonic 1.4" that is turned by Servo Motor of Hi-tech Company in ±90° vertical and ±180° horizontal. One of the other ability of this cam-

era is to zoom very well. Operator can control this pan tilt to find victims in environment.
But always have a 360 degree view of surrounds is very useful for robot driving. So we place a 360 degree view camera with Full-Circle Lens Overview of SONY Company in center of robot that gives us better view of surrounds.
The Third camera is the very small CCD camera with 105 degree view for the front of robot. There are some pictures of our camera include: CCD Panasonic camera and SONY camera:

Fig. 48 Panoramic Camera

Camera	CCD	CCD 1.3" (380 Line TV)
	Resolution	500H x 5827 (0.5 Lux - f2.0)
		>46 dB (1/100000 sec – 127 cc – 120 mA)
	وزن	200 gr
	Dimension	Ø87 x 72 mm
Motor	Motor	Servo HS700BD (Hitec)
	قدرت	Couple 80 N.cm (0.19 s/60°)
	Volt	6V
	Dimension	58.5 x 51 x 28 mm
	وزن	102 gr
	Price	63.60 €

Fig. 49 Zoom camera

Fig. 50 Front camera

8. Robot Locomotion

First let's talk about mobility of locomotion systems to have a clear view for judging.

8.1. Comparing the mobility of systems

The Q question in this section is that, how can we compare the mobility of dynamic structures. We have used "the mobility index comparison method" for comparing. First we introduced our mechanical design parameters:

- **Size**
- **Efficiency**
- **Environmental parameters**
  - Thermal
  - Ground Cover
  - Topography
  - Obstacles
- **Complexity**
- **Speed and Cost**

Designing functions for comparing mobility are:

- Step/Elevation Area: *Negotiable step height divided by the elevation area of mobility system*
- Step/System Height: *Highest negotiable wall or platform, whichever is shorter, divided by mobility system height*
- Crevasse/System Length: *Negotiable crevasse width divided by vehicle length (in the case of variable geometry vehicles, the shortest length of the mobility system)*
- System Width/Turning diameter: *Vehicle width divided by outermost swept diameter of turning circle*

- System Width/Turning-Around-a-Post Width: *Vehicle width divided by width of path it sweeps when turning around a very thin post*
- Ground Pressure

By using comparison functions introduced at above we found that track system is suitable for our aim. So we describe about track system notions.

8.2. Track mobile system

There has long been a belief that tracks have inherently better mobility than wheels and anyone intending to design a high mobility vehicle should use tracks. While tracks can breeze through situations where wheels would struggle, there are only a few obstacles and terrains which would stop a six wheeled rocker bogie vehicle, but not stop a similar sized tracked vehicle. They are

- very soft terrain: loose sand, deep mud, and soft powder snow
- obstacles of a size that can get jammed between wheels
- crevasses

They get this higher mobility at a cost of greater complexity and lower drive efficiency, so tracks are *better* for these situations, but not inherently better overall.

8.2.1. Kinds of track construction methods

Track systems are made up of track, drive sprocket, idler/tension wheel, suspension system, and, sometimes, support rollers. There are several variations of the track system, each with its own set of both mobility and robustness pros and cons.

- The design of the track itself (steel links with hinges, continuous rubber, tread shapes)
- Method of keeping the tracks on the vehicle (pin-in-hole, guide knives, V-groove)
- Suspension system that supports the track on the ground (sprung and unsprung road wheels, fixed guides)
- Shape of the one end or both ends of the track system (round or ramped)
- Relative size of the idler and/or drive sprocket

There are also many varieties of track layouts and layouts with different numbers of tracks. These various layouts have certain advantages and disadvantages over each other.

- One track with a separate method for steering
- The basic two track side-by-side
- Two tracks and a separate method for steering
- Two track fore-and-aft
- Several designs that use four tracks
- A six-tracked layout consisting of two main tracks and two sets of flipper tracks and each end

8.2.2. Kinds of track shape

Tracks shapes shown below:

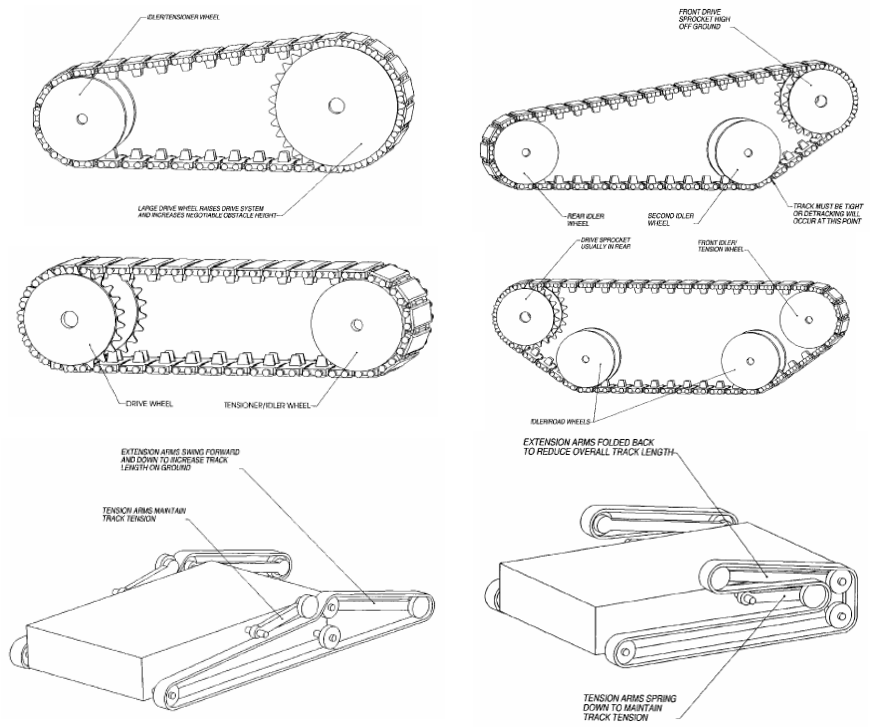


Fig. 51 Track shapes

8.2.3. Track suspension system

The space between the drive sprocket and idler wheel needs to be uniformly supported on the ground to achieve the maximum benefit of tracks. This can be done in one of several ways. The main differences between these methods are drive efficiency, complexity, and ride characteristics.
For especially long tracks, the top must also be supported, but this is usually a simple passive roller or two evenly spaced between the drive sprocket and idler. The main types of ground support methods are

- Guide blades
- Fixed road wheels
- Rocker road wheel pairs
- Road wheels mounted on sprung axles

The most complex, efficient, and smooth ride is produced by mounting the road wheels on sprung axles. There are three main types of suspension systems in common use.

- Trailing arm on torsion spring
- Trailing arm with coil spring
- Leaf spring rocker

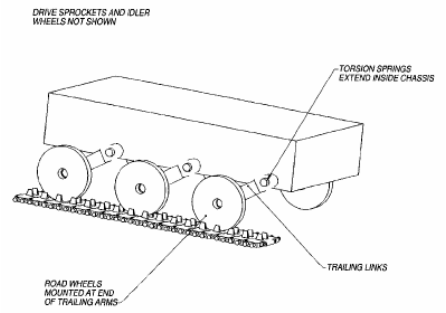


Fig.. Trailing arm on torsion spring

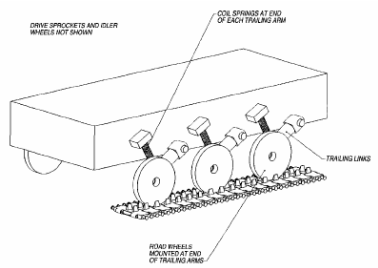


Fig.. Trailing arm with coil spring

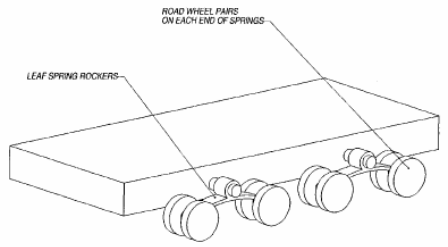


Fig. 52 Leaf spring rocker

8.3. Designing procedure

According to introduction, Arian III is a four-tracked active joint robot that its flippers can rotate separately at 90deg/sec. Although the shape of this robot is typical, their performances are extensively various; so the distinctions may not be appeared at a glance. Arian III robot design procedure is done with optimization aim.

To simplify the design process and save design optimization time, most of designs have been done in parametric form.

In design process, the theatrical calculations are done by MATLAB software then the results used in SOLID WORKS designing software, which collaborates with WORKING MODEL 4D and NASTRAN FEM analyzer software. So any changes in

entry of designing m.files lead to dimensional changes in output and causes variations at element dimensions in solid parts dimensions. After this step the changes manually checked at FEM software.

This example illustrates rear pulley design process step by step:

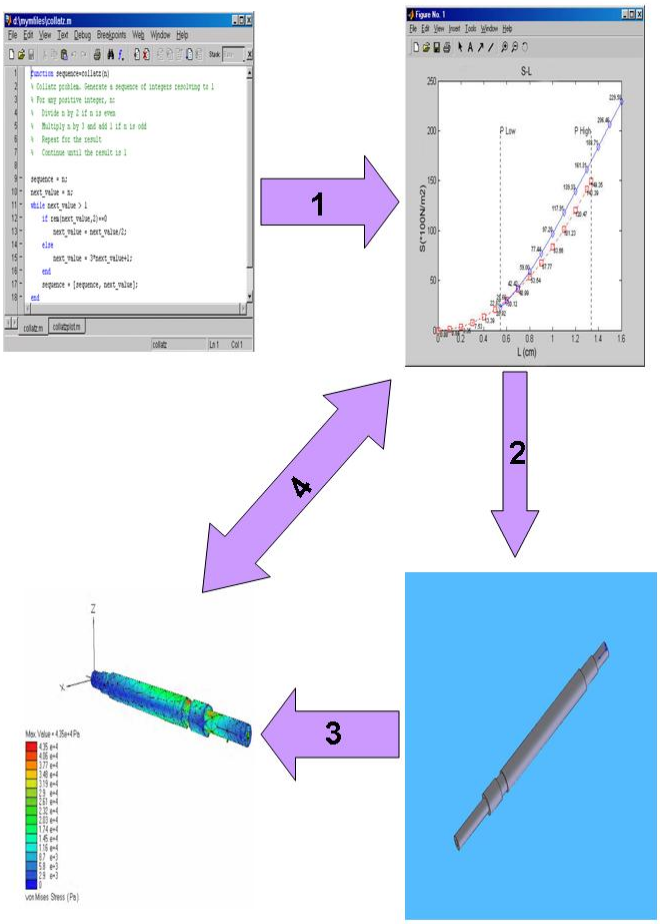


Fig. 53 Design cycle

stage 1 shows calculating methods and software outputs; stage 2 expresses automatic modeled part in SOLID WORKS software;stage3 shows manual analyzing in NASTRAN software; stage 4 shows comparing and optimizing of modeled part

8.3.1. Base calculation of Arian III designing

This section presents analysis of Robot locomotion system based on synchronous belts analysis.

Traditional understanding of timing belt drives comes from power transmission applications. However, the loading conditions on the belt differ considerably between power transmission applications and conveying and linear positioning applications.

- **Cinematic of problem (main body & flippers):**

$$d = \frac{p.z_p}{\pi}$$

d: *pitch diameter*
p: *nominal pitch*
z_p***:*** *number of pulley teeth*

$$d_o = d - 2u = \frac{p.z_p}{\pi} - 2u$$

d_o***:*** *outside diameter of pulley*
u*: pitch differential*

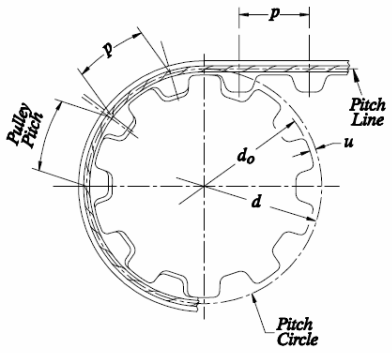


Fig. 54a. Belt and pulley mesh for inch series and metric T-series, HTD and STD series geometry.

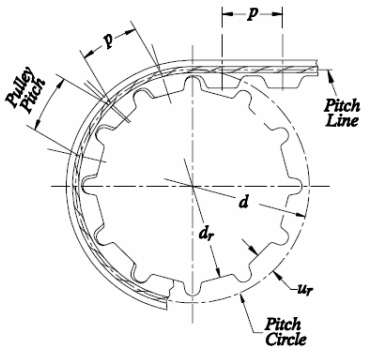


Fig. 54b. Belt and pulley mesh for AT series geometry.

Inch pitch and metric T series belts are designed to ride on the top lands of pulley teeth, the tolerance of the outside pulley diameter may cause the pulley pitch to differ from the nominal pitch (see Fig. 1a). On the other hand, metric AT series belts are designed to contact bottom lands (not the top lands) of a pulley as shown in Fig. 1b. Therefore, pulley pitch and pitch diameter are affected by tolerance of the pulley.

$$d_r = d - 2u_r = \frac{p.z_p}{\pi} - 2u_r$$

dr: *root diameter*

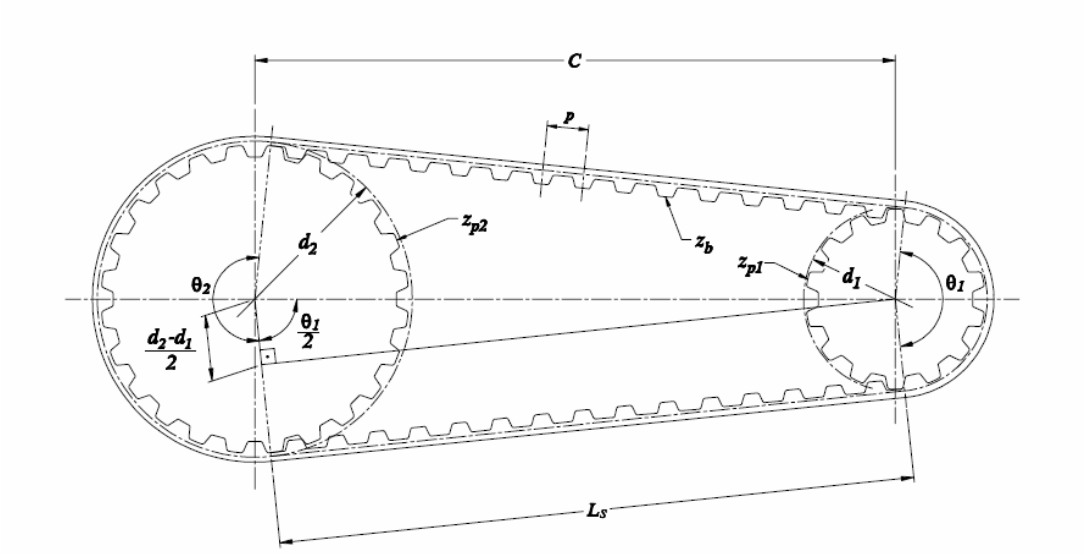


Fig.55 Belt drive with unequal pulley diameters.

$$L = p.z_b$$

L: *belt length*

$$\theta_1 = 2\arccos\left(\frac{d_2 - d_1}{2.C}\right)$$

θ_1: *angle of warp around the small pulley*
d_1: *pitch diameter of small pulley*
d_2: *pitch diameter of large pulley*

$$\theta_2 = 2\pi - \theta_1$$

θ_1: *angle of warp around the large pulley*

$$L_s = C.\sin\left(\frac{\theta_1}{2}\right)$$

L_s: *span length*

$$L = 2.C.\sin\left(\frac{\theta_1}{2}\right) + \theta_1.\frac{d_1}{2} + (2\pi - \theta_1).\frac{d_2}{2}$$

Since θ_1 is a function of C does not have a closed form solution for C. It can be solved using any of available numerical methods. An approximation of the center distance as a function of the belt length is given by:

$$C \approx \frac{Y + \sqrt{Y^2 - 2.(d_2 - d_1)^2}}{4}$$

$$\text{where } Y = L - \frac{\pi.(d_2 + d_1)}{4}$$

Kinetic of problem (main body & flippers):

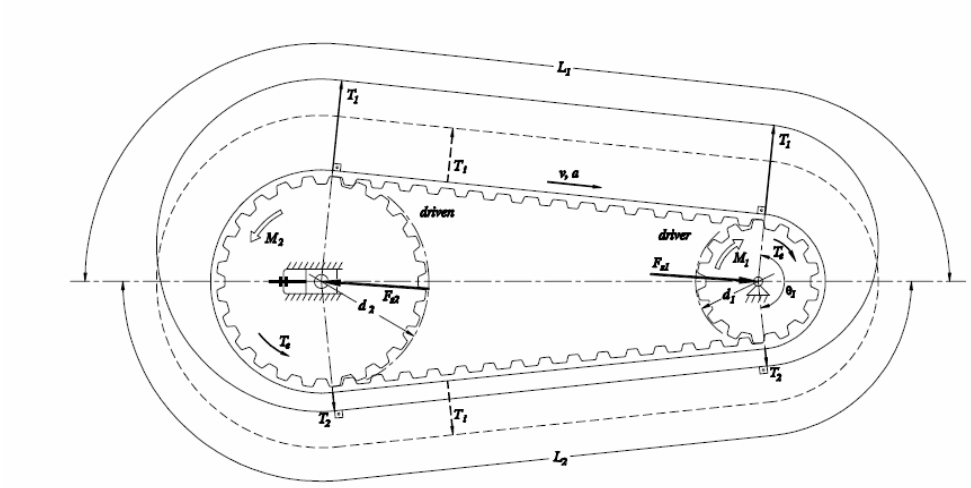


Fig. 56 Power transmission and rotary positioning.

During operation of belt drive a difference in belt tensions on the entering (tight) and leaving (slack) sides of the driver pulley is developed. It is called *effective tension*, ***Te***, and represents the force transmitted from the *driver* pulley to the belt

$$T_e = T_1 - T_2$$

T_e: *effective tension*
T_e: *tight side tension*
T_e: *slack side tension*

$$M = T_e . \frac{d}{2}$$

M: *driver torque*

$$M_1 = T_e . \frac{d_1}{2} = \frac{M_2}{\eta} . \frac{d_1}{d_2} = \frac{P_2 . d_1}{\eta . \omega_2 . d_2} = \frac{P_2}{\eta . \omega_1}$$

M_1: *driver torque*
M_2: *driven torque*
η: *system efficiency*
P_2: *driven power*

$$\omega_2 = \omega_1 . \frac{d_1}{d_2}$$

$$\omega_{1,2} = \frac{\pi . n_{1,2}}{30}$$

ω_1: *driver angular speed (rad/sec)*
ω_2: *driven angular speed (rad/sec)*
n: *angular speed(rpm)*

$$T_e = F_a + F_f + F_w + F_g + F_{ab} + F_{ai}$$

$$F_a = m_R . a$$

F_a: *acceleration force*
M_R: *robot mass*

$$F_f = \mu_r . m_R . g . \cos\beta + F_{fi}$$

F_f: *friction force*
μ_r: *dynamic coefficient of friction*
F_{fi}: *load independent resistance (seal drags, preload resistance, viscous resitance,etc.)*

$$F_g = m_R . g . \sin\beta$$

F_g: *weight of robot parallel to the inclined plane*

$$F_{ab} = \frac{w_b.L.b}{g}.a$$

F_g: *inertial force to accelerate belt*

$$F_{ai} = \frac{2.J_i.\alpha}{d}.a = \frac{m_i}{2}.\left(1 + \frac{d_b^2}{d^2}\right).a$$

F_i: *inertial force to accelerate idler pulley*
m_i: *idler mass*
J_i: *idler inertia*
d: *idler diameter*
d_b: *idler bore diameter*

F_w: *external force*

Next step:

$$F_f = \mu_r.\sum_{k=1}^{n_c} N_k = \mu_r.\cos\beta.\sum_{k=1}^{n_c} W_k = g.\mu_r.\cos\beta.\sum_{k=1}^{n_c} m_k$$

n_c: *number of contacted dents*
N_k: *partial weight of robot parallel to the inclined plane*
N_k: *partial mass of robot*

The presented calculations were used in designing of Arian III (see following models)

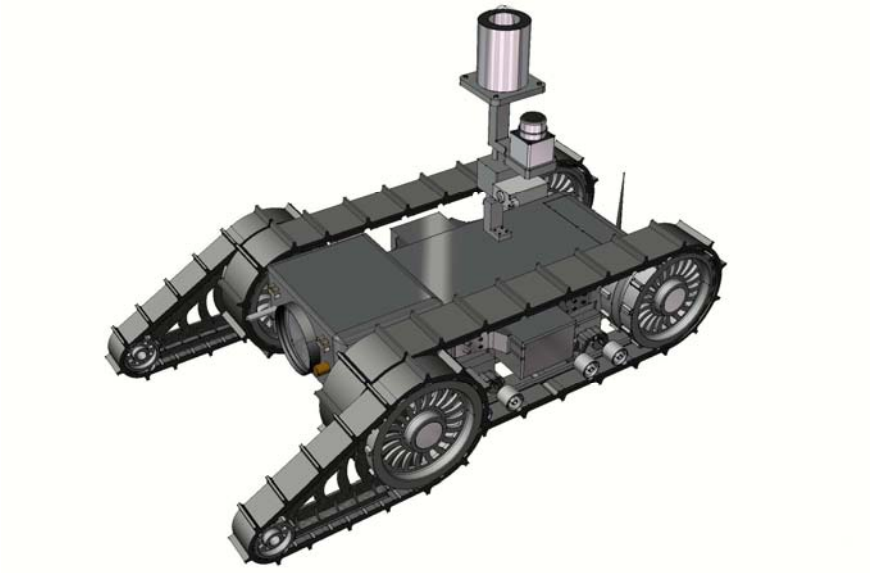


Fig. 58 Isometric view of Arian III

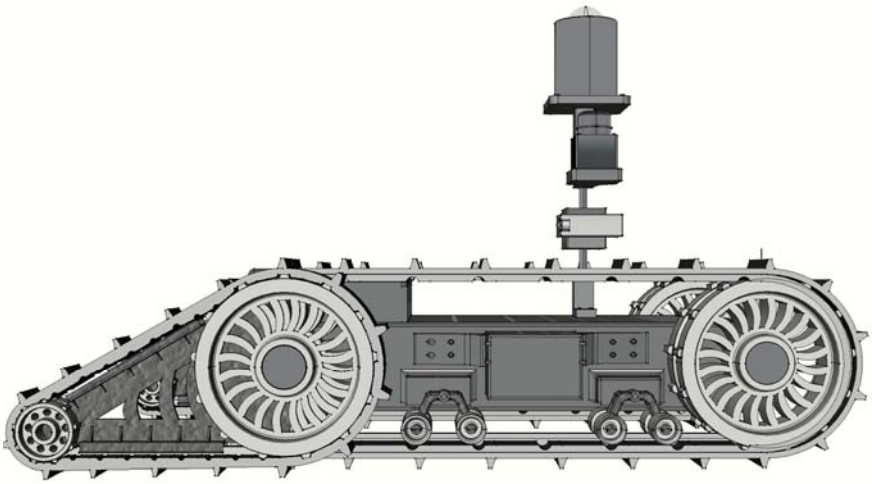


Fig.59 Lateral view of Arian III

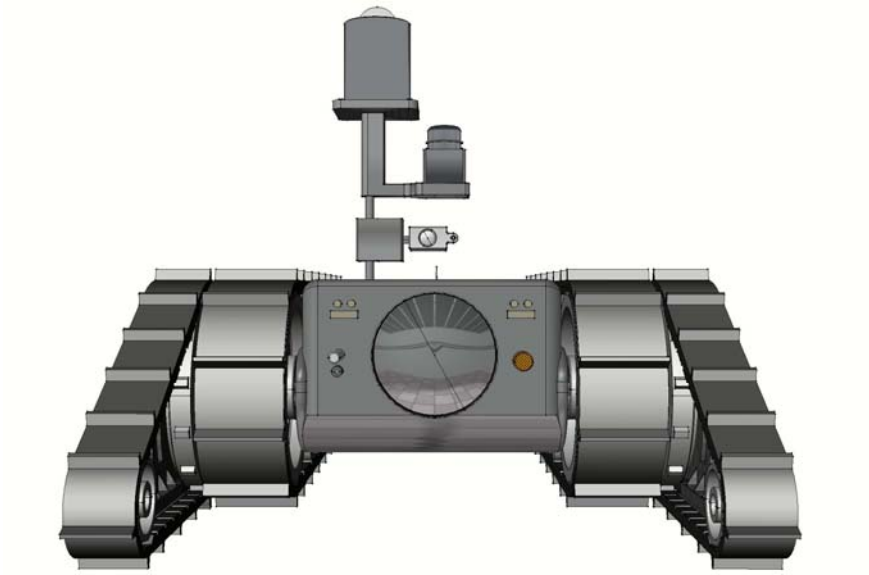


Fig.60 Front view of Arian III

8.3.2. Design Simplification

In order to simplification of usage and improving 'fool prevention' we have exerted our extreme efforts which are named below:

- **Packing power transmission system**
  PT package contains DC motors, all of bearings, gears, shafts as a robust package. By using this feature the operator can repair PT and replace tracks simply.
- **packing power supply**
  For decreasing the wasted time at battery charging process, in Ariana robots, batteries are fitted in a water proof unit which is connected to the body by a single MPC connector.

8.3.3. Robot Stability

Distinct characteristic of four track robots is their capability of increasing stability of system. But by this mechanism the stability of system is increased just in one direction (along robot length) so the weak point remains in opposite direction (along width). to have access for controlling stability in two directions, Arian III benefits two separately controlled fillipers.

9. Other Mechanisms

9.1 Power Supply

We have two ways for supply the electrical energy in our robot:

1- **Battery Energy**
2- **Solar Cells**

9.1.1 Battery Energy

The basic power source of our robots supply is Nimh Cells.

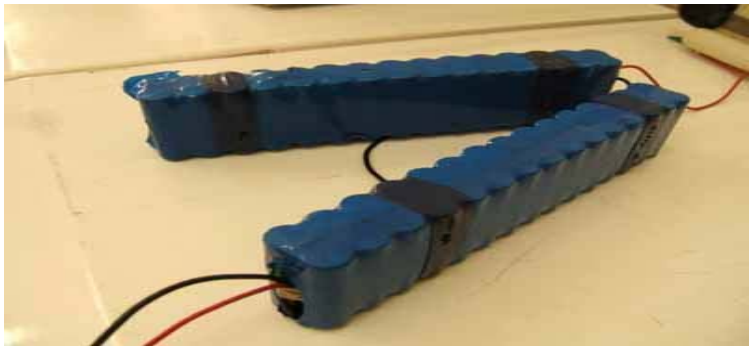


Fig. 61 Battery packages

We have two battery packs and each of them have 30 cells of batteries and 14V and 9A current. Each cell has 3000 mAh capacity.
There is very important subject about robot power supply is to be easy for recovery or charging.
For realize this purpose we must have 2 cases:

1) easy plug and unplug
2) fast charging

At the first we consider the nearest place for plug and unplug the batteries between the tracks.
At the second we made the special fast battery charger for charge of our packages in Maximum rate of fast charge that we allow for the cells.
We consider many condition or solution for the find end of charging. This condition is Temperature of packages, voltage of packages, current of charging and charge time. Time is last reason for finish the charging. With this charger we can charge all of our battery packages less then of 2 hour.

9.1.2 Solar Cells

Since our goal of this project has been make of an industrial robot. This robot must be can work in real condition and in this condition may we don't access to Electrical energy directly.

Therefore we use the one Kind of solar cells with name of Roll able solar cells or Portable power films.

Fig. 62 Solar Cell

This Kind of cells is flexible and waterproof.
We used of 2 Roll of P3-55watt in parallel mode therefore we have 110 watt Power.
Dimension and Weight has been show in below:

Table. 6

Dimensions and Weight	
Folded	
Length, in (mm)	11 (279)
Width, in (mm)	9 (229)
Thickness, in (mm)	1.3 (33)
Deployed	
Length, in (mm)	55 (1397)
Width, in (mm)	32 (813)
Thickness, in (mm)	0.1 (2.5)
Weight	
Weight, lb (kg)	4.4 (2)
Power to weight ratio watt/lb (watt/kg)	12.5 (27.1)

Electrical characteristics have been show in below:

Table. 7

Electrical Characteristics*	
P3 Product Number	P3-55
Nominal System Voltage Rating	16V
Nominal Rated Power (watts) at STC	55
Nominal Rated Voltage at MPP (VDC) at STC	20
Nominal Rated Current (amps)	2.8
Open Circuit Voltage (volts)	30
Short Circuit Current (amps)	3.7
Thermal Characteristics	
Power (%/C)	-0.5
Voltage (%/C)	-0.5
Cell Temperature Operating Range	-40° F to 176° F/ -40° C to 80° C

Volt Amp chart has been show in below:

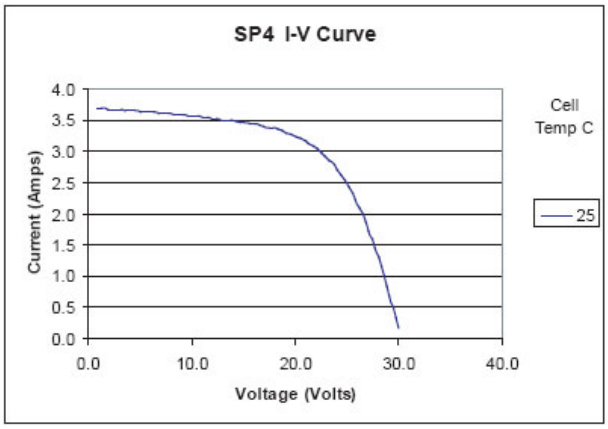


Fig. 63 SP4 I-V Curve

9.2. Heads up Goggles

As mention on the previous parts we use a heads up goggles for alerting the critical information to the operator. We use a MD-6 model, a MicroOptical's heads up goggles. Some information about it mention below:

Working in conjunction with your current patient-monitoring system, MicroOptical's MD-6 Critical Data Viewer displays vital signs where you need them most — right before your eyes. Without obstructing your natural field of vision, the viewer duplicates the live display of your monitor as a floating image positioned a few feet in front of you — displaying vitals in real-time. By keeping both patient and critical data in your hand-eye axis, the viewer allows you to view vital signs repeatedly without having to look away at a monitor. By connecting to two VGA sources, surgeons can view two sets of critical data, such as vital signs and cath lab images, and alternate between them. Compact, lightweight and comfortable, it easily attaches to surgical or prescription eyewear. Experience patient monitoring at the next level with the Surgical Data Viewer System from MicroOptical.

MD-6 Critical Data Viewer*	
Display Format	640 x 480, 60 Hz refresh rate, landscape
Color Depth	18-Bit (262,144 colors)
Optics Location	Left or right eye configurable
Field of View	Approximately 16° horizontal, 20° diagonal
Focus Range	Adjustable focus from 2 to 15 feet
Input Signal	VGA, SVGA, XVGA, 60-75 Hz
Input Connector	Std. VGA, male DB-15 connector, 18"cable and 25' ext. cable
Power Requirements	2 W
Power Source	7.2 V lithium ion rechargeable battery
Continuous Operation	5 hours on full charge
Viewer Weight	1.25 ounces (35 grams)
Cable Length	4 feet from the viewer to the belt-worn controls/battery
Eyeglass Frame Size	Quick release mount fits many std. eyeglasses/safety glasses
Operating Temperature	32 to 140°F (0 to 60°C)
Storage Temperature	-4 to 140°F (-20 to 60°C)

9.3. Body state sensors

For the better control of the Arian III we need to know about the body state of robot. At this time we take advantage of three sensors: the first is 3DM sensor which determines the spatial orientation of the Arian III. The second sensor is strain gage sensor and the third is absolute encoder for determining the state of flipper.

9.3.1. Strain gage sensor

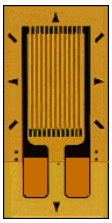
As you know strain gage sensors work on differentiation of resistant characteristics. Arian III benefits feature in two categories:

i) Optimization and control the friction force between flipper tracks and the ground

ii) Determining the lateral force inserted to the flipper body structure

To reach this aim we have used two uniaxial strain gages (in circumferential and lateral manner) on flippers body. With circumferential strain gage we can measure normal force of surface, which leads to obtain friction force. And with the other one, lateral force is obtained which is used to automatic protection of flipper power transmitter shaft.

The important point in this section is determination of critical point for strain both lateral and circumferential. We developed optimized software in MATLAB to reach this aim.(see figure x)



Typical uniaxial strain gage pattern designed to measure strains in the direction of the gridlines. Gage lengths for Micro-Measurements strain gages range from 0.008 in to 4.000 in (0.20 mm to 101.6 mm).

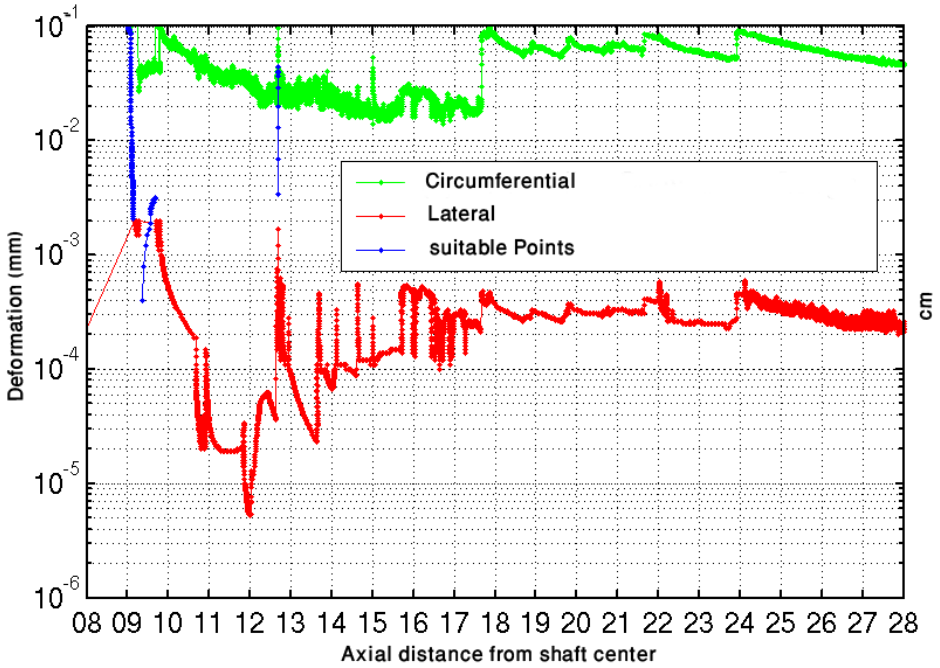


Fig.64 simulation graph in MATLAB

9.3.2 Absolute Encoder

For determining the state of the flippers relating to the body of Arian III we use a absolute encoder. In fact by using this absolute encoder we determine the flippers angle relating to the body. This sensor model is AC36, an ACURO-industry's encoder and has these features:

- Overall length 36 mm
- For equipment engineering and industry

- Up to 17 Bit singleturn and 12 Bit multiturn
- Hollow shaft 6 mm
- +100°C operating temperature
- 10 000 rpm continous operation
- Optical encoder with a true geared multiturn
- SSI or BiSS interface
- Option Sine 1 Vpp
- 500 kHz bandwidth

The AC 36 is an absolute optical encoder with a true geared multiturn, optical sensing technology and 36 mm diameter. Equiped with a solid-shaft the AC 36 is mechanical compatible with all common inkremental encoders. The compact design allows to replace the adequate incremental encoders directly. As a result the technical facilities of absolute encoders can be used for the first time in equipment engineering and also in medical engineering. The mechanical design consists of two ball bearings supported mechanical shaft assembly. The AC 36 complements the ***ACURO-industry*** series with small frame sizes and the same performance as 58 mm versions.

BiSS-Interface
Unique within his class the AC 36 provides fully digital position data up to 17 bit (singleturn) and 12 bit (multi-turn) over the bidirectional synchronous interface with a variable clock rate up to 10 MHz. This corresponds to a singleturn resolution of more than 130 000 measured steps.Backward compatibility is realized through the SSI interface together with 2048 sine-cosine periods per revolution.

Integrated diagnostic system
The AC 36 is based on latest OptoAsic technology with an advanced diagnostic concept. A continuous plausibility check controls the internal signal processing for each increment. A code check guarantees that the encoder signal represents bit by bit the measured rotation. Also the operating temperature of the encoder can be measured, read out and monitored over warn and alarm bits with 8 bit resolution (1°C). Monitoring and controlling of the operating temperature ensures a maximum lifetime of the LED. Eventual failures are indicated early over warn bits.

Housing diameter	37.5 mm
Protection class shaft input	IP 64
Protection class housing	IP 64
Flange	pilot flange
Shaft diameter	6 mm
Max. speed	Continuous 10 000 min^{-1}, short term 12 000 min^{-1}
Starting torque	0.01 Nm
Moment of inertia	2.5 x 10^{-6} kgm^2
Shock resistance DIN EN 60068-2-27	1 000 m/s^2 (6 ms)
Vibration resistance DIN EN 60068-2-6	100 m/s^2 (10 ... 2 000 Hz)
Operating temperature	-25...+100 °C
Storage temperature	-15...+85 °C (because of packing)
Weight approx. ST/MT	80 g / 130 g

Supply voltage	5 V, -5 % /+10% or DC 7 - 30 V
Max. current w/o load ST/MT	50 mA / 100mA
Interface	Standard SSI or BiSS
Lines / drives	Clock and Data / RS422
Output code	Gray or binary
Resolution singleturn	13 Bit - 17 Bit
Resolution multiturn	12 Bit
Incremental signals optional	Sine - Cosine 1 Vpp
Number of pulses	2048
3dB limiting frequency	500 kHz
Alarm output	Alarm bit (SSI Option) warning bit and alarm bit (BiSS)
Connection	Cable axial or radial

10. Team Training for Operation (Human Factors)

Experiences in designing and implementation of previous versions of our robotic products (Arian I, II) imply us that independency of a system to the human factors is itself a great factor and must be reduced; so the newest version of Ariana equipped with a simple and user friendly interface.
Therefore, the method of introducing information is more important than its own characteristic; then Arian III benefits heads up goggles, showing essential data (e.g. coordination, speed … of robot and vital signs of victim) and touch panel LCD is used to simplicity of handling the menus of control unit (see fig.65)

Fig. 65 Control unit of Arian II

Joy stick used in control unit of Arian III introduces position whilst the others give orientation.

Fig. 66 Joy stick System

11. Possibility for Practical Application to Real Disaster Site

By rough analysis of latest rescue robot competitions, we found innovative ideas that work properly in controlled and unreal conditions, but in real world we are faced with several undesirable situations. For example mechanical and thermal shocks, environmental noises, x-ray radiations… influence on robot capabilities. As mentioned, all Ariana robots have been made with these regards. The following pictures show Arian I stair climbing at Ariana Home (Shahed Research Center).

Fig. 67 Ariana I stair climbing

And night mission of Ariana II at Tehran suburb (Chitgar).

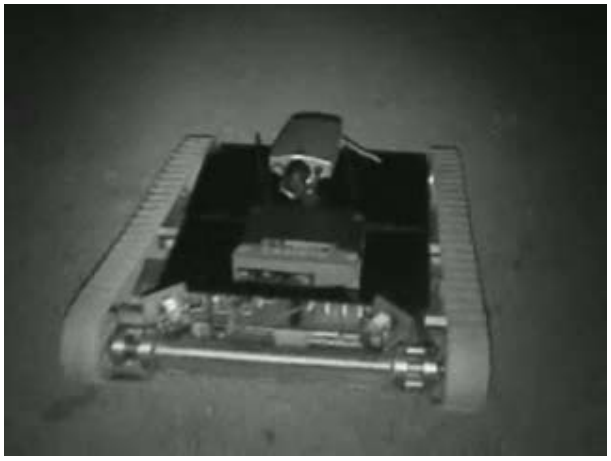


Fig. 68 Ariana II night mission

Arian III continued this way as our group policy, for example drop limit of Arian II (0.8 m) has improved to about 2 meters, its operating time improved from 4 hours in his elder brother to 8 hours and equipped with some new electrical devices like thermal and omni cameras…

12. System Cost

Category	Part Number	Product Name	Company	Cost
Navigation	Devantech	Ultra Sonic	SFR-08	50$
Navigation	Devantech	Elec. Compass	CMPS03	40$
Navigation	Microstrain	Gyro Sensor	3DM-G	1800$
Navigation	Lassen iQ	GPS	GPSTM	100$
Navigation	Sony	Panoramic Camera	---	900$
Victim Sensor	Sony	Microphone	ECM-HS1	95$
Victim Sensor	Sony	Speaker	Simple 8 ohm	10$
Victim Sensor	---	Thermal Camera	---	20000 $
Victim Sensor	Omega	IR Thermometer	---	300$
Victim Sensor	Valtronics	CO2 Sensor	---	1000$
Victim Sensor	GKS	CCD Camera	---	480$
Power Supply	---	Battery	12V 18A	200$
Power Supply	---	Solar Cell	12V 2A	2 x 750$
Monitor	Micro Optic	Heads-up Goggle	MD-6	1200$
Simple Sensor	---	Strain gage Sensor	---	30$
Simple Sensor	ACURO-ind.	Absolute Encoder	AC36	500$
Locomotion	Faulhaber	Motor	3557	2x400 $
Locomotion	Faulhaber	Gearbox	38/1s	2x300 $
Locomotion	Faulhaber	Shaft Encoder	HEDS558	2x150 $
Mechanical Main Body and Manipulation	---	---	---	1500$
Mechanical Mobility	---	---	---	600$
Mechanical Parts	---	---	---	600$
Total Price				**31705 $**

References

1. Paul E. Sandin, “Robot Mechanisms and Mechanical Devices”, The McGraw-Hill Companies, Inc. 2003

2. “Timing Belt Theory”, Mectrol Corporation, . April 2001

3. United States Fire Administration and National Fire Association. *Rescue Systems I*, 1993.

4. J. Casper, M. Micire, and R. Murphy, Issues in Intelligent Robots for search and Rescue, *SPIE Ground Vehicle Technology II*, 4: 41-46, 2000.

5. Cameron, J.M. and Arkin, R.C., Survival of Falling Robots, SPIE *Vol 1613 Mobile Robots VI*, 1991.

6. Arkin, R.C., Survivable Robotic Systems: Reactive and Homeostatic Control, *Robotics and Remote Systems for Hazardous Environments*, ed. M. Jamshidi, Prentice-Hall, 1993.

7. Packbot, iRobot Corporation: www.packbot.com

8. Dragon Runner, CMU, National Robotics Engineering Consortium,: http://www.rec.ri.cmu.edu/projects/dragon/index.shtml

9. nBot Balancing Robot, David Anderson: http://www.geology.smu.edu/~dpa-www/robo/nbot/

10. South. D.W. and Mancuso, J.R., 1994, *Mechanical Power Transmission Components*, Marcel Dekker, Inc., New York.

11. Rao, S.S. 1995, *Mechanical Vibrations*, Addison-Wesley Publishing Company, Reading, MA.

12. Beer, F.P, and Johnston, E.R., *Vector Mechanics for Engineers: Dynamics*, 1988, McGraw-Hill, New York.

13. C. Lundberg, H. I. Christensen, A. Hedstrom , “The Use of Robots in Harsh and Unstructured Field Applications”, Centre for Autonomous Systems (CAS), Numerical Analysis and Computer Science (NADA), Royal Institute of Technology (KTH)

14. Brian Yamauchi and Pavlo Rudakevych ,“Griffon: A Man-Portable Hybrid UGV/UAV,”,
Industrial Robot, vol. 31, no. 5, pp. 443-450, 2004.

15. Andreas Hedstr¨om, Henrik I Christensen, and Carl Lundberg, “A Wearable GUI for Field Robots”
Centre for Autonomous Systems (CAS), Numerical Analysis and Computer Science (NADA), Royal Institute of Technology (KTH), S-10044 Stockholm, Sweden,

16. R. Simmons, L. Henriksen, L. Chrisman, G. Whelan, “Obstacle avoidance and safeguarding for a lunar rover”, *Proc. AIAA Forum on Advanced Developments in Space Robotics*, Madison WI, August 1998.

17. L. Matthies, A. Kelly, T. Litwin, G. Tharp, “Obstacle detection for unmanned ground vehicles: a progress report”, *Robotics Research: The Seventh International Symposium*, G. Giralt and G. Hirzinger (eds), Springer-Verlag, 1996. 8. M. Hebert, R. MacLachlan, P. Chang, “Experiments with driving modes for urban robots”, Proc. SPIE Conference on Mobile Robots, Boston MA, September 1999.

18. Y. Xiong and L. Matthies, “Vision-guided autonomous stair climbing”, *Proc. IEEE International Conference on Robotics and Automation (ICRA),* San Francisco, April 2000.

19. A. Kelly, “A 3D State Space Formulation of a Navigation Kalman Filter for Autonomous Vehicles”, Techinical Report, CMU-RI-TR-94-19, Robotics Institute, Carnegie Mellon University, May, 1994.

20. M. H. Grewal, L. R. Weill, A. P. Andrews, *Global Positioning Systems, Inertial Navigation, and Integration*, John Wiley & Sons, New York, 2001. 4. M. H. Hebert, C. Thorpe, A. Stentz, A. Kelly, *Intelligent Unmanned Ground Vehicles, Autonoums Navigation Research at Carnegie Mellon*, Kluwer Academic Publishers, Massachusetts, 1997.

21. J. Albus, H. McCain, and R. Lumia, “Nasa/nbs standard reference model for telerobot control system architecture (nasrem),” Tech. Rep. NBS Technical Note 1235, Robot Systems Division, National Bureau of Standards, Gathersburgh, VA, 1987.

22. R. G. Simmons, “Structured control for autonomous robots,” IEEE Transactions on Robotics and Automation, vol. 10, no. 1, pp. 34–43, 1994.

23. R. C. Arkin, “Integrating behavioral, perceptual, and world knowledge in reactive navigation,” Robotics and Autonomous Systems, vol. 6, pp. 105–122, 1990.

24. R. A. Brooks, “A robust layered control system for a mobile robot,” IEEE Journal of Robotics and Automation, vol. RA - 2, pp. 14 – 23, March 1986.

25. M.W.M.G. Dissanayake, P. Newman, H.F. Durrant-Whyte, S. Clark, and M. Csorba. An experimental and theoretical investigation into simultaneous localisation and map building. *Experimental Robotics IV*, pages 265–274, 2000.

26. D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 2000.

27. J.S. Gutmann and K. Konolige. Incremental mapping of large cyclic environ ents. In *Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 318–325. IEEE, 2000.

28. B.J. Kuipers and Y.T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systmes*, 8(1-2):47–63, 1991.

29. J.J. Leonard and H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–82, 1991.

30. M. DeGroot, Probability and Statistics, 2nd Edition, Addison-Wesley Publishing: Massachusetts, 1986.

31. Courtney, J. and Jain, A. "Mobile robot localization via classi¯cation of multisensor maps", Proc. of the IEEE Int. Conf. on Robotics and Automation, pp. 1672-1878, 1994.

32. Hartigan, J., "A k-means clustering algorithm", Applied Statistics, **28**, pp. 100-108, 1979.

33. L. Kleeman, "Optimal estimation of position and heading for mobile robots using ultrasonic beacons and dead-reckoning", Proc. of the IEEE Int. Conf. on Robotics and Automation, **3**, pp. 2582-2587, Nice, 1992.

34. Leonard, J. and Durrant-Whyte, H. Directed sonar sensing for mobile robot navigation, Kluwer Academic Plublishers, Norwell, MA, 1992.

35. Schiele, B. and Crowley, J. "A comparison of position estimation techniques using occupancy grids", Robotics and autonomous systems, **12**, pp. 163- 171, 1994

36. Betke, M.,&Gurvits, L. (1993). *Mobile robot localization using landmarks* (Tech. ep. SCR94TR474). Princeton: Siemens Corporate Research.
Borenstein, J. (1987). *The ursing robot system*. Doctoral dissertation, Technion, Haifa, Israel.

37. Borenstein, J., Everett, B., & Feng, L. (1996). *Navigating Mobile Robots: Systems and Techniques*. Wellesley, MA: A. K. Peters, Ltd.

38. Buhmann, J. (1995). Data clustering and learning. In M. Arbib (Ed.), *Handbook of brain theory and neuralnetworks* (pp. 278–282). Cambridge, MA: Bradfort Books/MIT Press.

39. Buhmann, J., Burgard, W., Cremers, A. B., Fox, D., Hofmann, T., Schneider, F., Strikos, J., & Thrun, S. (1995). The mobile robot Rhino. *AI Magazine*, *16*, 31–38.

40. Borenstein, J. and Feng. L, 1995a, "UMBmark: A Benchmark Test for Measuring Odometry Errors in Mobile Robots." To be presented at the SPIE Conference on Mobile Robots, Philadelphia, Oct. 22-26.

41. Borenstein J. and Feng. L, 1995b, "Measurement and Correction of Systematic Odometry Errors in Mobile Robots." Accepted for publication in the *IEEE Journal of Robotics and Automation*, May 1995.

42. Arleo, A. and Gerstner, W. (2000) Spatial cognition and neuro-mimetic navigation: A model of hippocampal place cell activity. *Biological Cybernetics*, 83: 287-299.

43. Chang H.J. & Freeman W.J. (1996) Parameter optimization in models of the olfactory system, *Neural Networks*, 9: 1-14.

44. Freeman, W.J. (1975) *Mass Action in the Nervous System.* Academic Press, 1975.

45. Freeman, W.J. (2000) *How Brains Make Up Their Minds*, Columbia University Press.

46. R. Cassinis, D. Grana, and A. Rizzi. Self-localization using an omni-directional image sensor. In International Symposium on Intelligent Robotic Systems, pages 215{ 222, July 1996.

47. J. A. Castellanos, J. M. Martinez, J. Neira, and J. D. Tardos. Simultaneous map building and localization for mobile robots: A multisensor fusion approach. In IEEE ICRA, pages 1244{1249, 1998.

48. F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In IEEE ICRA, pages 1322{1328, 1999.

49. H. Durrant-Whyte, M. Dissanayake, and P. Gibbens. Toward deployment of large scale simultaneous localization and map building (SLAM) systems. In Proc. of Int. Simp. on Robotics Research, pages 121{127, 1999. [6] A. Georgiev and P. K. Allen. Vision for mobile robot localization in urban environments. Submitted to IEEE IROS, 2002.