Noname manuscript No.
(will be inserted by the editor)

Modular Design and Implementation for a Sensory-Driven Mobile Manipulation Framework

Ayssam Elkady · Vipul Babariya · Jovin Joy · Tarek Sobh

the date of receipt and acceptance should be inserted later

Abstract A mobile manipulator is a manipulator mounted on a mobile platform with no support from the ground. We are already in the process of building a platform (RISCbot II) which consists of a comprehensive sensor suite and significant end-effector capabilities for manipulation. In order to reduce the uncertainty in localization, sensor fusion is used to create an efficient and effective user interface to facilitate teleoperation by enhancing the quality of information that is provided to the teleoperator.

We are developing a framework for a modular design of sensory modules, actuation platforms, and task descriptions that will be implemented as a tool to reduce and streamline efforts in designing robotic platforms. The framework is comprised of three modules. The first module encapsulates the sensors which gather information about the remote, or local, environment. The second module defines the platforms and actuation methods. The last module describes the tasks that the platforms will perform such as teleoperation, navigation, obstacle avoidance, manipulation, 3-D reconstruction, and map building.

This paper presents the modular design process of the RISCbot II mobile manipulator. In the design process, the overall design of the system is discussed and then the control process of the robot is presented. Furthermore, the tasks that the RISCbot II can perform such as teleoperation, navigation, obstacle avoidance, manipulation, and face detection and recognition are described.

Keywords Mobile Manipulation · Middleware · Manipulability · Teleoperation · Face Detection · Face Recognition · Modular Design.

1 Introduction

A mobile manipulator offers the dual advantage of mobility offered by a platform, and dexterity offered by a manipulator. In addition, the mobile platform extends the

Ayssam Elkady
School of Engineering - University of Bridgeport, 221 University Avenue Bridgeport, CT 06604, U.S.A. Tel.: +860-303-9536
Fax: +203-576-4766
E-mail: aelkady@bridgeport.edu

manipulator's workspace. Mobile manipulators are potentially useful in dangerous and unpredictable environments such as construction sites, aerospace, underwater, service environments, and nuclear power stations.

Many ideas and implementations have been proposed and applied to autonomous mobile robots such as Stanford Research Institute International (SRI) AI Center's Saphira [1], Carnegie Mellon University (CMU) Robot Institute's Teambots [2], the Royal Institute of Technology's BERRA [3], CLARAty ([4], [5], [6]), and CotsBots [7].

Sensor fusion has been an active area of research in the field of computer vision and mobile robotics. Sensor fusion is the combination of sensory data from different sources, resulting in better and more reliable information on the environment than data derived from any individual sensor. Sensor fusion algorithms are useful in low-cost mobile robot applications where acceptable performance and reliability is desired, given a limited set of inexpensive sensors like ultrasonic and infrared sensors. Sensor fusion has some critical problems including the synchronization of sensors. Different sensors have different resolutions and frame rates so the sensors need to be synchronized before their results can be merged by fusing the data from multiple sensors and presenting the result in a way that enables the tele-operator to perceive the current situation quickly. Sensor fusion is used to reduce operator workload, enabling him or her to concentrate on the task itself. Sensor fusion is commonly used to reduce uncertainty in localization, obstacle avoidance, and map building. Furthermore, sensor fusion may be used to improve teleoperation by creating user interfaces which efficiently facilitate understanding of remote environments and improve situational awareness.

We have constructed a mobile manipulator platform called RISCbot II (the prototype of the RISCbot II is shown in Figure 1). The RISCbot II mobile manipulator has been designed to support our research in algorithms and control for the autonomous mobile manipulator. The objective is to build a hardware platform with redundant kinematic degrees of freedom, a comprehensive sensor suite, and significant end-effector capabilities for manipulation. A demo for RISCbot II is shown in [8]. Furthermore, we are developing the *RISCBot II middleware* which resides between the operation system and the software applications. The purpose of this middleware is to manage the heterogeneity of the hardware, improve software application quality, simplify software design, and reduce development costs.

In this paper, rather than describing a mobility/manipulability platform, we concentrate on the modular design process for the mobile manipulation paradigm as applied to a construction task for the platform. We introduce the proposed software and middleware architecture and the tasks that the RISCbot II is performing; specifically teleoperation, navigation, obstacle avoidance, manipulation, face detection, and facial recognition.

2 Modeling of the RISCbot II

In order to specify the position of the robot on the plane, we establish a relationship between the global reference frame of the plane and the local reference frame of the robot. The origin O of the global reference frame is arbitrary selected on the plane, as shown in Figure 2. The point C is the center of mass of the robot. The origin P of the local reference frame of the robot $\{X_p,Y_p\}$ is at the center of the robot. The basis defines two axes relative to P on the robot chassis and is thus the robot's local reference frame. The position of P in the global reference frame is specified by coordinates x and

Fig. 1 A prototype of the RISCbot II .

y, and the angular difference between the global and local reference frames is given by θ. The pose of the robot is described by a vector ξ.

$$\xi = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{1}$$

To describe robot motion, it is necessary to map the motion along the axes of from the global reference frame to the robot's local reference frame. This mapping is accomplished using the orthogonal rotation matrix $R(\theta)$.

Where $R(\theta) = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$

This orthogonal rotation $R(\theta)$ is used to map the motion ξ in the global reference frame to motion ${}^{P}\xi$ in terms of the local reference frame $\{\mathrm{X_p}, \mathrm{Y_p}\}$. This operation is:

$${}^{P}\zeta = R(\theta)\zeta \tag{2}$$

As described in [9], the velocity of point P and C in the global reference frame $\dot{\xi}_p$ are described in equations 3 and 4 . The three kinematic constraints can be written in the form of equation 5.

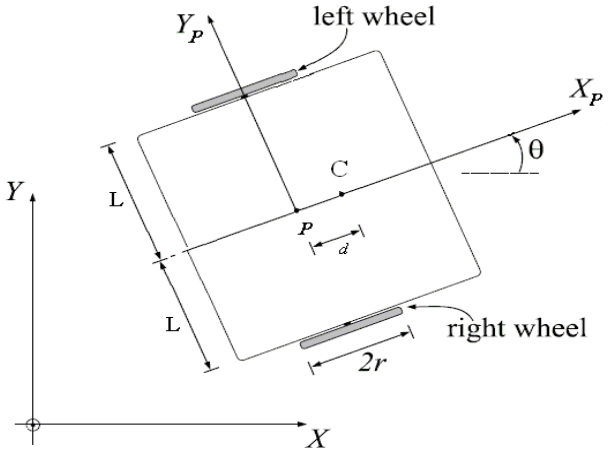

Fig. 2 Kinematic model of the RISC.

$$\dot{\xi}_p = \frac{r}{2} \times \begin{bmatrix} (\dot{\Phi}_l + \dot{\Phi}_r)cos(\theta) \\ (\dot{\Phi}_l + \dot{\Phi}_r)sin(\theta) \\ \frac{(\dot{\Phi}_r - \dot{\Phi}_l)}{l} \end{bmatrix} \tag{3}$$

$$\dot{\xi}_c = \begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\theta} \end{bmatrix} = \frac{r}{2} \times \begin{bmatrix} (\dot{\Phi}_l + \dot{\Phi}_r)cos(\theta) + \frac{d}{l}(\dot{\Phi}_l - \dot{\Phi}_r)sin(\theta) \\ r(\dot{\Phi}_l + \dot{\Phi}_r)sin(\theta) + \frac{d}{l}(\dot{\Phi}_r - \dot{\Phi}_l)cos(\theta) \\ \frac{(\dot{\Phi}_r - \dot{\Phi}_l)}{l} \end{bmatrix} \tag{4}$$

$$A(q)\dot{q} = 0 \tag{5}$$

Where $A(q) = \begin{bmatrix} sin\theta & -cos\theta\ d & 0 & 0 \\ cos\theta & sin\theta & l & -r\ 0 \\ -cos\theta & -sin\theta\ l & 0 & r \end{bmatrix}$ and $\dot{q} = \begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\theta} \\ \dot{\Phi}_r \\ \dot{\Phi}_l \end{bmatrix}$ Where $q = [x_c, y_c, \theta, \Phi_r, \Phi_l]^t$

2.1 Kinematic Position Control

The objective of a kinematic controller is to follow a trajectory described by its position or velocity profile as a function of time. Consider the coordinate transformation into polar coordinates, with its origin at the goal position.

$$\rho = \sqrt{\Delta x^2 + \Delta y^2} \tag{6}$$

$$\beta = \tan^{-1}(\frac{\Delta y}{\Delta x}) \tag{7}$$

$$\alpha = \beta - \theta \tag{8}$$

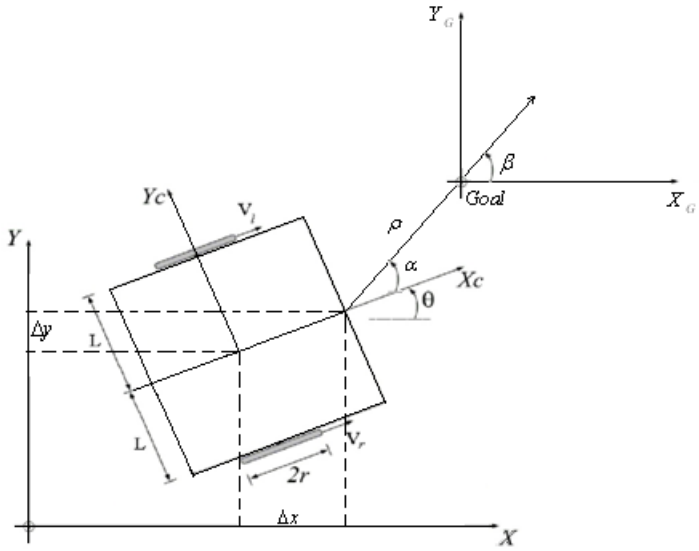

Fig. 3 RISCbot II Kinematics.

α and β are the angles shown in Figure 3 and ρ is the distance between the center of the robot's wheel axle and the goal position. This yields a system description in the new polar coordinate, using matrix equation 9.

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} -\cos\alpha & 0 \\ \frac{\sin\alpha}{\rho} & -1 \\ \frac{\sin\alpha}{\rho} & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{9}$$

The control signals must now be designed to drive the robot from its actual configuration to the goal position. When the linear control law in equation 10 is combined with equation 9, a closed-loop system results [equation 11].

$$\begin{aligned} v &= K_p \rho \\ \omega &= K_\psi \psi + K_\alpha \alpha \end{aligned} \tag{10}$$

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} -K_p \rho \cos\alpha \\ K_p \rho \sin\alpha - K_\psi \psi - K_\alpha \alpha \\ K_p \rho \sin\alpha \end{bmatrix} \tag{11}$$

The closed-loop control system [equation 11] is locally exponentially stable if $K_p > 0$, $K_\psi < 0$ and $K_\alpha - K_p > 0$, which is proved in [10].

3 Design structure of RISCbot II

In the structural design, as shown in Figure 4, the three layer design was maintained. The base is comprised of plywood, holding square core iron structure to provide support

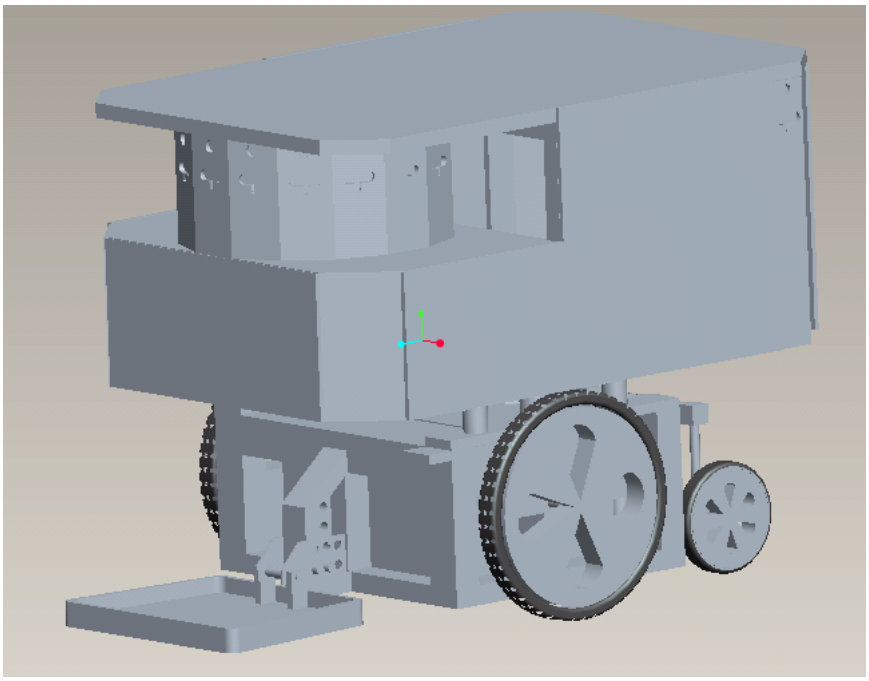

Fig. 4 Final Prototype Structure.

for the manipulator. A cylindrical structure to mount sensors is fixed on the central layer. The topmost layer is used to mount a stereo camera and a manipulator. The manipulator is mounted in such a way as to transfer its weight to the square core iron from the base.

Furthermore, a 2” diameter hole is drilled on the bottom layer to pass the wirings from the battery. The joystick of the wheelchair is mounted on the core iron structure vertically. There are seven sensors mounted on the back face of the bottom plywood and the center of all sensors is kept in one line. Different views of the structure are shown in Figure 5. Plywood was chosen in the final design for building the layered structure due to its cost effectiveness and easy availability.

3.1 Individual Parts Description

A summary of individual parts used in the final prototype is described in table 1.

3.2 Bottom Plywood Sheet

The bottom plywood sheet used has a dimension of 32” × 22” × 0.75”, as shown in Figure 6. In order to mount all components, the thickness of the plywood sheet was set at 0.75”. The front pocket is used to mount the laptop, as shown in Figure 7. This plywood sheet is directly mounted on six bosses extending from the wheelchair frame. Two I-support and central support structures are mounted on top of the sheet.

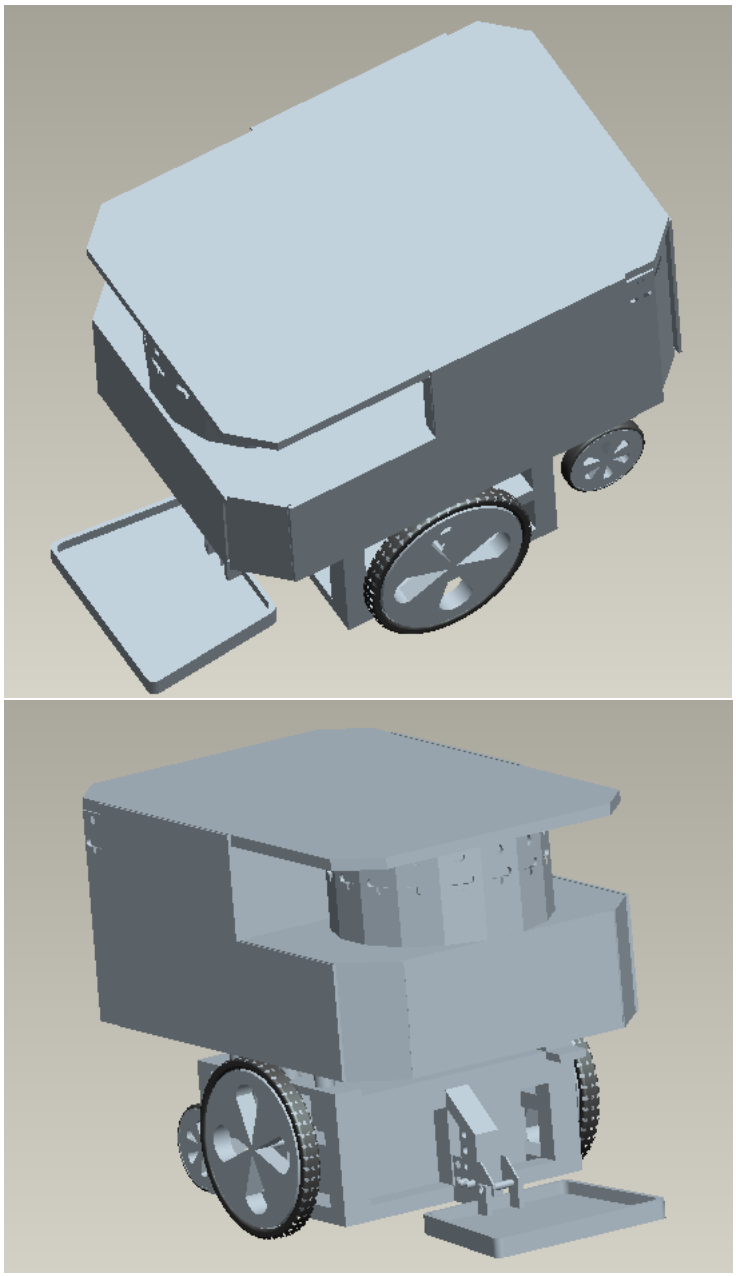

Fig. 5 Different views of the final prototype structure.

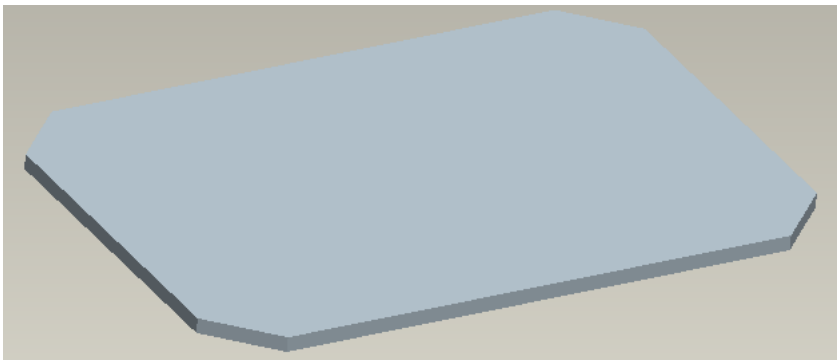

Fig. 6 Bottom plywood sheet.

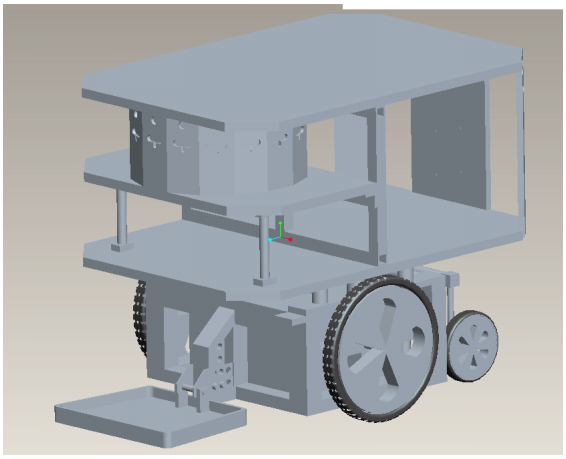

Fig. 7 Final prototype showing structural support.

3.3 Middle Plywood Sheet

This plywood sheet, shown in Figure 8, is mounted on the square brackets of the I-support and central support structures, extending out of the bottom plywood sheet, as shown in Figure 7. Its dimensions are 16” $\times$ 22” $\times$ 0.75”. This sheet is used to support the cylindrical sensor mount.

Table 1 Individual parts used in the final prototype.

Part	Material Type	Quantity
Top Plywood Sheet	Plywood	1
Middle Plywood Sheet	Plywood	1
Bottom Plywood Sheet	Plywood	1
Central Support Structure	Core Iron	1
Support Part	Core Iron	4
Cylindrical Part	Wood	1
Backward Sensor Part	Wood	3

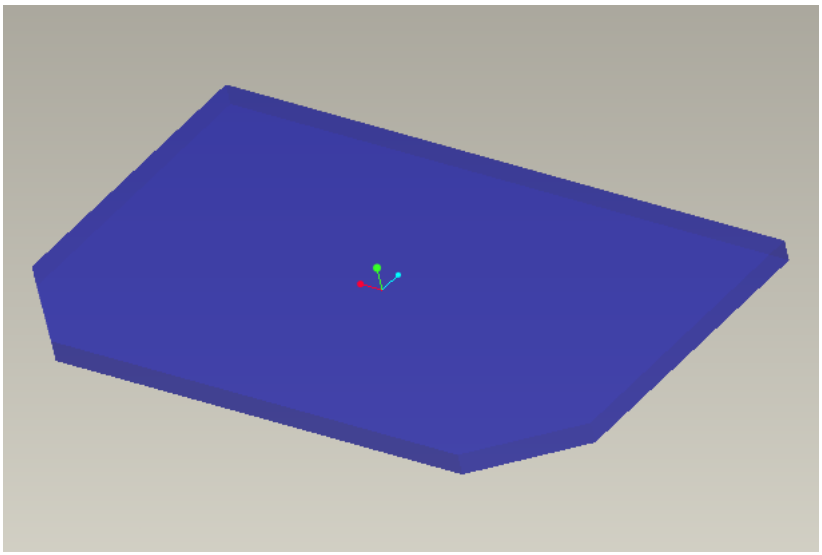

Fig. 8 Middle Plywood sheet.

3.4 I-Support Part

The I-support part is manufactured from core iron (0.5” diameter) with a square bracket welded on each end, as shown in Figure 9. Four of these are used in the design; two of them are used to support the middle plywood sheet and the others are mounted on the middle plywood to support the top plywood, as shown in Figure 10.

3.5 Cylindrical Sensor Mount

This structure is assembled from nine rectangular wooden plates arranged around an incomplete circle (1.25π) of radius 9.5”. Each rectangular plate is 2.75” thick and mounts two types of sensors, namely, infrared and ultrasonic, as shown in Figure 11.

3.6 Central Support Structure

This structure is a rectangular hollow frame made of 1” thickness core iron. In the middle, an angle bracket is welded to support the middle plywood sheet. This support structure rests on the bottom plywood sheet and provides support for the top plywood sheet and manipulator.

3.7 Top Plywood Sheet

The top plywood, shown in Figure 13, is mounted on top of the central support structure. It is also supported by two I-supports extending from the middle plywood, cylindrical sensor mount and the rear sensor rack. This sheet supports the manipulator and stereo camera.

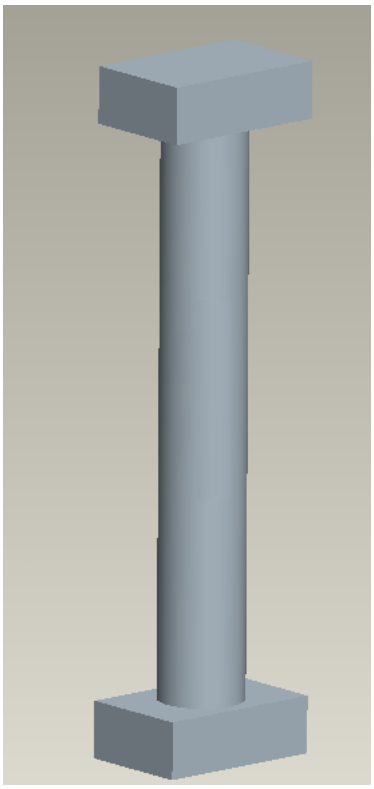

Fig. 9 I-Support Part.

3.8 Analysis of the RISCbot II Structure

The analysis of the structure is performed on ANSYS Workbench 11.0. A static structural analysis is performed to check stress, strain and deformation of the structure. The stress distribution in the structure is shown Figure 14. A maximum stress of 2.0086 kPa is measured on the edges of plywood. The strain generated in all the assembled components can be viewed in Figure 15. The total deformation found throughout the structure from the simulation is uniform, as shown in Figure 16. The maximum equivalent Elastic Strain and Total deformation found are negligible.

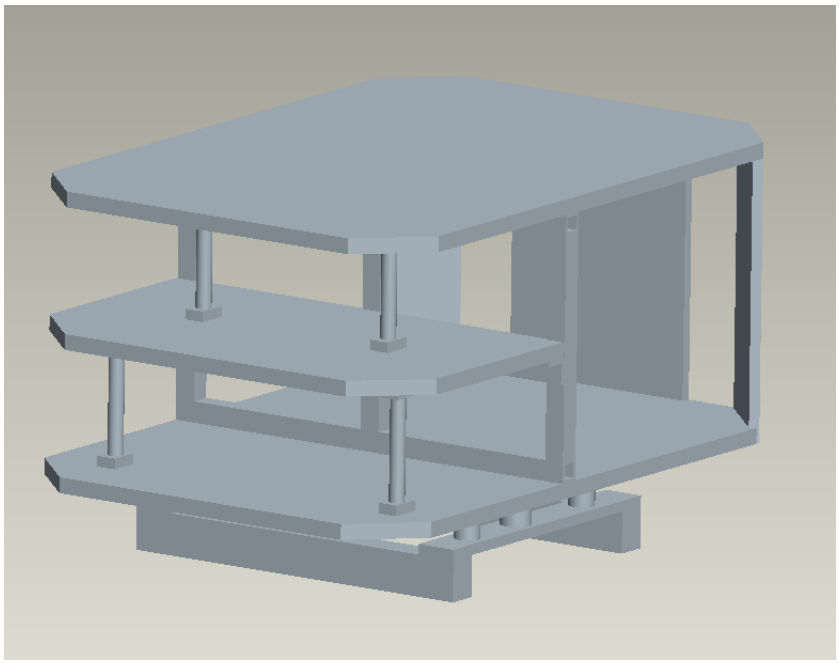

Fig. 10 Final prototype showing structural support without Cylindrical Sensor Mount.

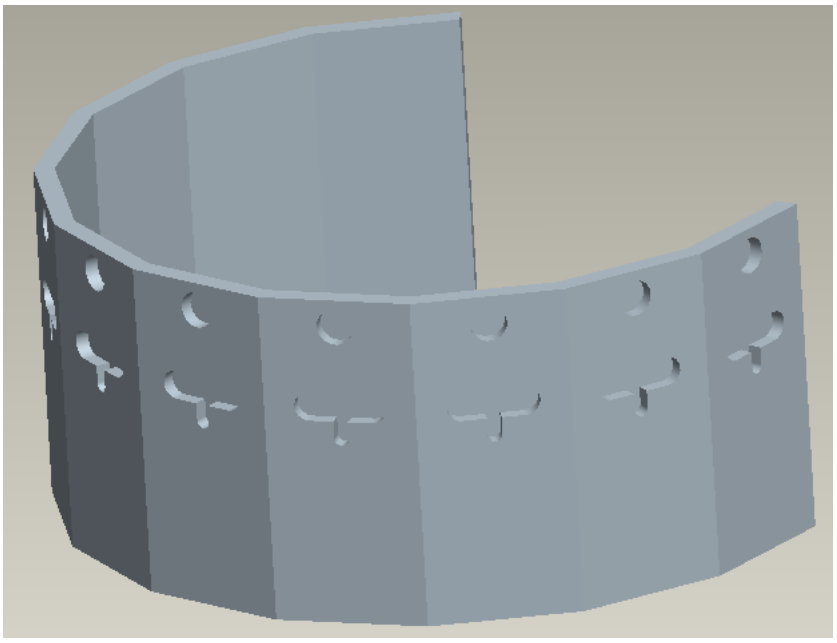

Fig. 11 Cylindrical Sensor Mount with rectangular plates on perimeter.

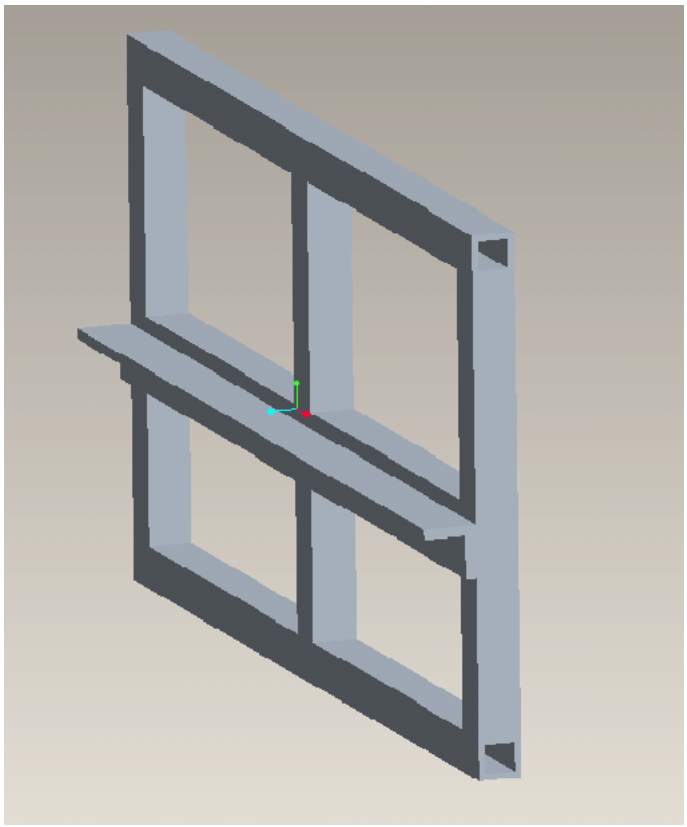

Fig. 12 Central Support Structure.

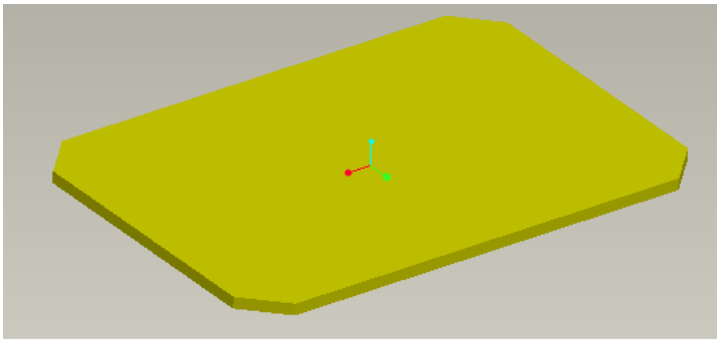

Fig. 13 Top Plywood Sheet.

4 RISCBot II Middleware Architecture

Lightweight RISCBot II middleware is an abstraction layer residing between the operation system and the software applications. It is designed to manage the heterogeneity of the hardware, improve software application quality, simplify software design, and

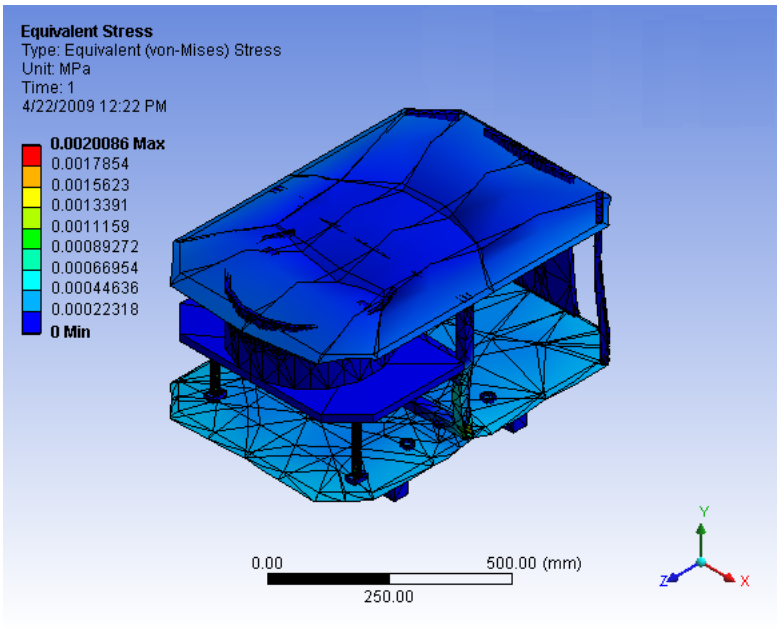

Fig. 14 Equivalent stress distribution throughout structure.

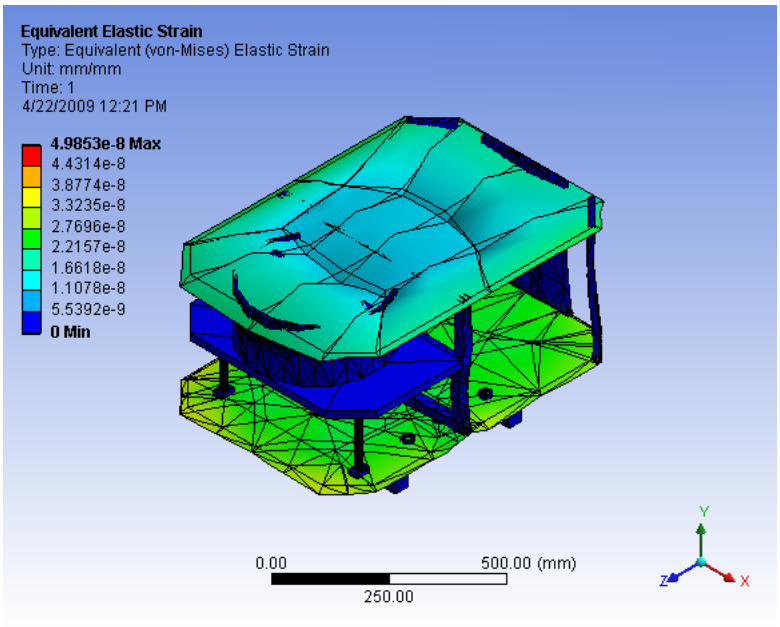

Fig. 15 Equivalent elastic Stain simulation result.

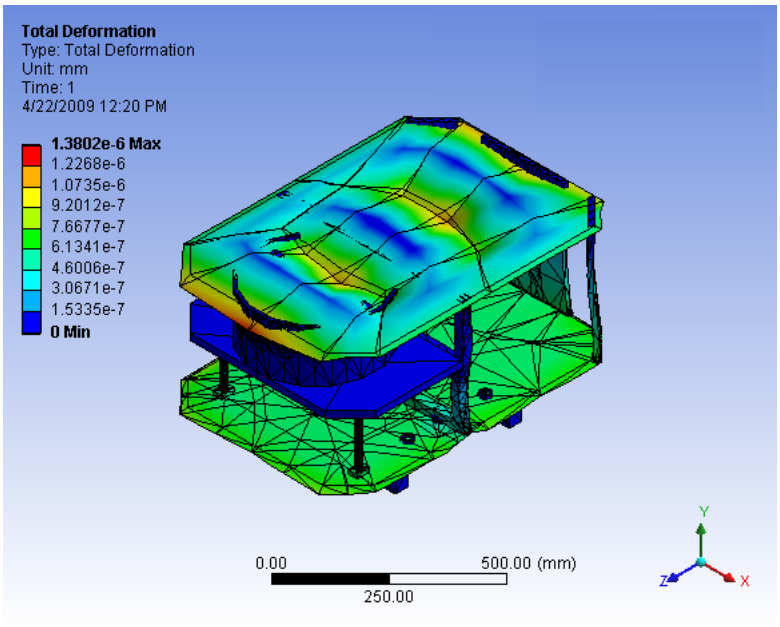

Fig. 16 Total deformation.

reduce development costs. A developer needs only to build his logic or algorithm as a component, after which his component can be combined and integrated with other existing components. Furthermore, if he wants to modify and improve his component, he needs only to replace the old one with the new one. Therefore, experiment efficiency will improve. Some advantages of RISCbot II software are:

- Real-time capabilities: RISCbot II software provides real-time capability because of the high speed communication between its modules.
- Schedule management: RISCbot II middleware supports time synchronization between cooperating modules.
- Plug and play is the key of the RISCbot II software, which allows dynamic configuration of the robot hardware and software.
- Software reuse: The framework provides the reusability of components/modules.
- Operating System independent.
- Network independent: The RISCbot II software supports various communication media and its model is an independent structure.

The architecture of the middleware has five primary layers for managing hardware, creating behaviors that will be used by many applications (as shown in Figure 17).

4.1 OS & Hardware Layer (OSHL)

The lower layer of robot middleware is the robot hardware system, composed of a variety of sensors, actuators and other hardware devices. It is also the operating system (i.e., Microsoft Windows, UNIX, etc.) that runs on the robot.

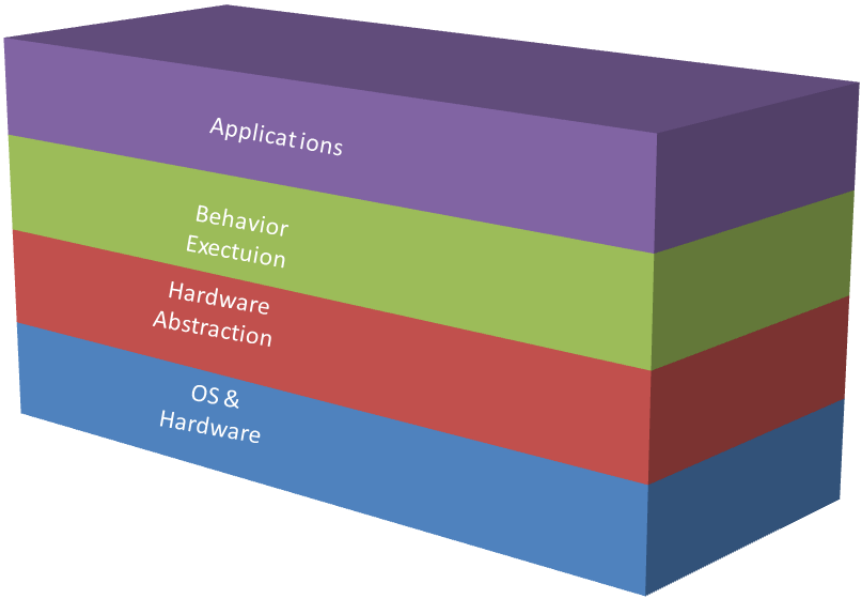

Fig. 17 The Architecture of the RISCbot II Middleware.

4.2 Hardware Abstraction Layer (HAL)

The Hardware Abstraction Layer is the robot control software composed of a variety of robot software components, control algorithms and services. Robot middleware hides the heterogeneity of lower hardware devices and provides the component interface for the upper layers call, independent of the robot hardware system. HAL removes hardware and operating system dependencies between the robot and the application. This assures portability of the architecture and application programs so that code for one robot can be used in later generations of products, which will extend engineering resources. The Hardware Configuration Descriptor (HCD), which is a XML descriptor, is used to describe the configurations and hardware properties. HAL also provides unified access to all network interfaces such that the upper layers do not become concerned with what function should be called when sending data to a remote or local module. For the purpose of consistency, the interface through which components can be accessed and controlled is standardized. The standard interface to hardware devices (sensors, and actuators) takes place through seven following operations: (All the operations will indicate error conditions if they fail.)

1. Open: Before using a device, it should be open. The purpose of the open call is to allow the system to initiate the device and fetch the attributes into the main memory for rapid access on later calls.
2. Close: When all accesses are completed, the devices are no longer needed and should be closed. Many systems encourage this by imposing a maximum number of open devices on processes.
3. Read. Data are read from a device. The caller must specify how much data are needed and must also provide a buffer in which to put them.
4. Write. Data are written to the device.

5. Get attributes. It returns the attributes and the configuration of a device.
6. Set attributes. Some of the attributes and the configuration can be changed.
7. Lock. Locking a device prevents multiple simultaneous accesses by different processes.

4.3 Behavior Execution Layer (BEL)

The Behavior Execution Layer (BEL) is comprised of modules and infrastructure used for sensing, decision-making, and autonomous action. This layer implements the logical link to the sensor or actuator. It provides access to the sensor data or actuation commands abstracted from the underlying physical connection of the resource. The layer hides the underlying sensor and actuator access mechanisms, e.g., communication protocols and hardware ports addressed in lower layers. The BEL understands the semantics of the behaviors requested by each application and efficiently generates new actions to fulfill the requirements of each behavior. The BEL has three types of execution modes:

- **Concurrent Execution**: When two or more behaviors run simultaneously in parallel, the BEL fuses the behaviors and removes any conflicts between them.
- **Time-sharing Execution**: Each behavior runs exclusively at one time. The time management system is implemented to schedule each behavior. The BEL interrupts running behaviors or resumes the suspended behaviors on schedule. The programmer is required to describe additional information that defines what is critical for the behaviors. The BEL must decide whether or not the current behavior can be suspended. When the robot suspends the current behavior, the BEL must automatically decide when the robot should resume it. When the robot does not suspend the current behavior, the BEL requires the robot to reject the new request or to store this request in a queue for requests awaiting execution.
- **Sequential Execution**: Each behavior runs using batch processing. The BEL inserts an interruption and follows the sequence to duplicate the details of actions.

4.4 Applications Execution Layer

This layer contains the application processes. They are provided with a set of functions for accessing the resources through the next lower layer. The layer uses interprosses communication (IPC) provided by the real time operating system to implement the client and server scheme. Clients can be consumers as well as suppliers. They can receive data from servers (the server distributes the sensor or internal data to all client processes sensors) or they can send data to the server (the server forwards the data to the appropriate receiver (e.g., the actuator)). This layer also accepts commands from the software applications and manages the behavior of a robot. The AEL provides the required application program interface (API) to the application layer such as installing/uninstalling a robot application, starting/stopping it, registering/unregistering a logical device, etc.

4.5 Applications

The system incorporates the use of algorithms such as SLAM, Obstacle avoidance, Navigation, vision, etc. We have already implemented some tasks including teleoperation, navigation, manipulation and obstacle avoidance. Every application component should implement its interface. This interface includes the functionalities of starting/stopping a resource, configuring the resource, and connecting a port of the resource to another resource's port. The Software Assembly Descriptor (SAD) describes a software configuration, the properties, and the connections among components.

4.6 RISCbot II Layer Architecture

Each layer consists of three components; the shared memory, the main process and the layer interface, as shown in Figure 18. The upper layer can access the lower layer through the layer application program interface (API) in the layer interface part. The purpose of the API is to awaken the main process to create the required (input, output or input/output) thread to deal with the request by sending or receiving data from the lower layer. The purpose of the main process is to handle the data flow to and from the layer API. On the other hand, it creates the input, output and input and output threads. The threads become active whenever new data are available from or for the lower layer, and sleep in the meantime. Shared memory is used for fast and efficient communications and should be protected by semaphores to avoid conflict. In shared memory, real and simulated data will be stored. The purpose of the simulated data is to offer the ability to simulate sensor readings and allow switching between two modes, simulation and real. This ability offers efficient testing and debugging of the application and also offers an intelligent auto-configuration advantage. Sending and receiving data may be synchronous (process data when available), periodic (whose period can be changed) or asynchronous (when some events are triggered). A synchronous message blocks a process till the operation is completed and the communicators are time synchronized. In synchronous message services, the deadlock should be considered. Asynchronous communication does not require that all parties involved in the communication be present at the same time. The asynchronous message system is used, for example, to notify the consumer about new data available in the shared memory.

5 Applications

5.1 Face Detection

Our primary method of detection is based on Haar-Link Features as described in [11] and OpenCV computer vision library. Haar encodes the existence of oriented contrasts between regions in images. A set of features can be used to encode the contrast exhibited by the human faces and their special relationships.

Our detection program is derived primarily from OpenCV and uses classifiers (a cascade of boosted classifiers working with haar-like features) that is trained with a few hundred sample views of human faces. Classifiers used in our project were obtained from the OpenCV library. The classifiers are built using positive and negative examples of the same size. After a classifier is trained it can be used to search across an image

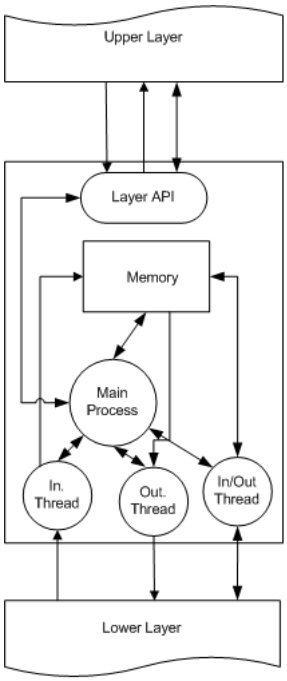

Fig. 18 The Architecture of a Layer in RISCbot II Middleware

at locations. The output of the classifier is 1 if it finds an object(face) in a region. The classifier is resized rather than the image to detect objects(faces) of different sizes. Therefore, to find faces of unknown size, the scan procedure is performed several times at different scales.

5.2 Face Recognition

Face Recognition is a task of identifying or verifying a person from a digital image or a video source. The primary method of face detection uses an overlapping energy histogram, as described in Face Recognition by Overlapping Energy Histograms[12]. For the sake of efficiency, OpenCV library was used extensively while developing the code.

The face Recognition algorithm is two step process with a training stage and search stage. The general approach is show in Figure 19. The training stage is performed initially to create a database; the database is not computed again as it is a time consuming process. The search phase uses the database created from the previous stage to look up unknown images. The overlapping energy Histogram is comprised of taking DCT coefficients of every 8x8 pixel value with a 75 percent overlap, as shown in Figure 20 (i.e Taking DCT for every 8x8 block after every 2 pixels). Once the image has been transformed into a histogram, Euclidean Distance is computed, as show in equation 12, where Ω is the feature vectors calculated via the histogram. Further; Euclidean Distance has been used in many face recognition techniques [12]. This distance is used to calculate the Database Threshold(θ), which is used for identifying and rejecting

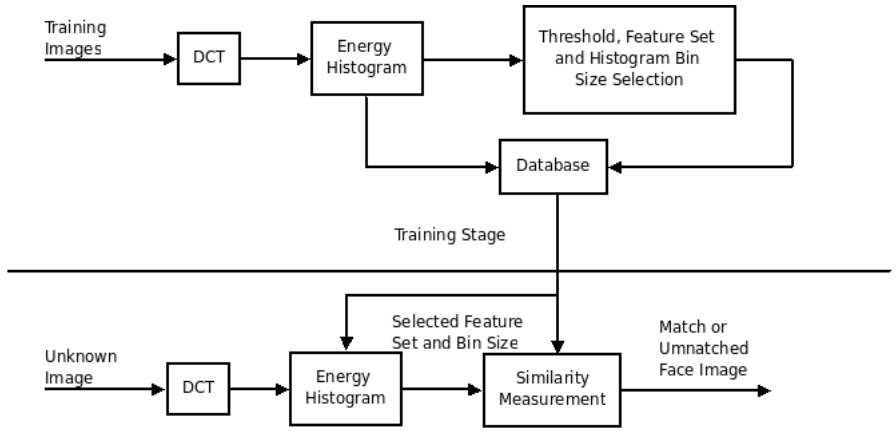

Fig. 19 Overview of Face Recognition Module

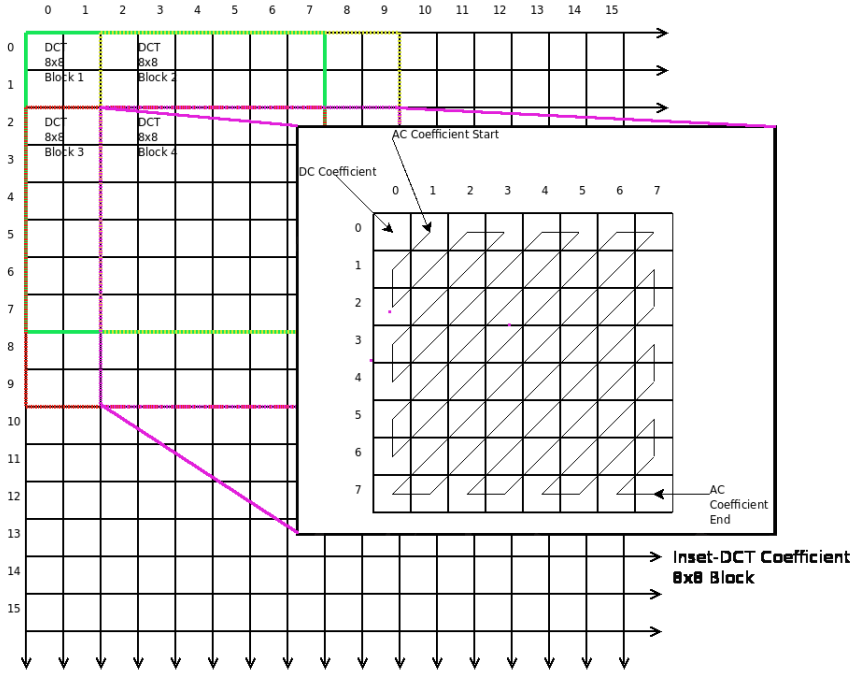

Fig. 20 Energy Histogram With 75 Percent Overlap

images at the search stage.

$$\epsilon_n = \|\Omega - \Omega_n\|^2 \tag{12}$$

5.3 Feature Extraction

Once the image with a face is transformed using overlapping DCT procedure values of Feature Set F2(F2=[DC,AC_{01},AC_{10},AC_{11}]), the values of F2 is put into a histogram. The real challenge has been in selecting the appropriate getting the appropriate bin sizes for calculating threshold values. The appropriate bin sizes was determined by running a series of test on known images at the time of development.

5.3.1 Threshold Selection

The threshold for the image database is calculated by intra and inter class information gained from the training dataset. Intra class(D) information is obtained via computing the Euclidean distance between the images of an individual. Inter class(P) information is obtained via computing the distance between images of an individual with others in the database. Once this is performed for all the images in the database, the Threshold(θ) is defined by

$$\theta = \frac{D_{max} + P_{min}}{2} \tag{13}$$

5.3.2 Image Search

Once the required feature from the image to be recognized is extracted, its features are cross computed with each individual in the database to see if it falls within the range of the database threshold. If more than one individual matches with the image that is searched for, the individual with the closest threshold value is usually the correct person.

5.3.3 Face Detection/Recognition Testing

Face Detection and Recognition modules have to be tested simultaneously with the input of Detection Program fed into the recognition program. The Face Detection program on an average detects up to five faces in real time (30 frames/second), running on a Dual Core Intel Processor, therefore bringing the total to 150 images/second. This is a 10 fold increase from the original Viola-Jones implementation. Figure 21 shows a sample output from the detection program. The main concerns regarding the number of images that can be detected per frame are the computational requirement and the need to maintain real time performance.

The recognition module, on the other hand, can take each of the detected faces and search the database to find possible matches. The initialization process of the Face Recognition database is found to be a processor hog, hence plans to recompute the database values at run time had to be abandoned. Another bottleneck is the total memory requirements for the database, which increases due to storing the feature vectors in uncompressed formats in system memory.

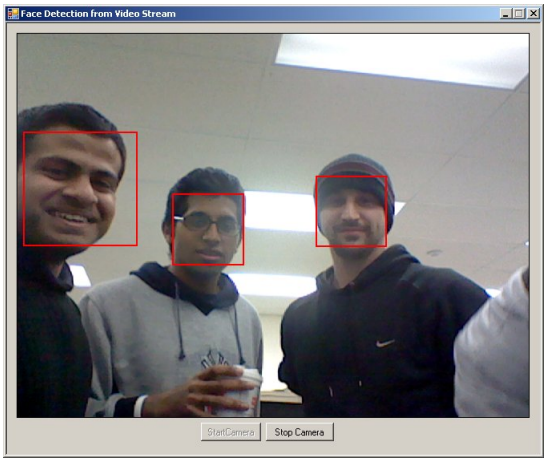

Fig. 21 Face Detection Module Detecting Multiple People in a Single Frame

5.4 Teleoperation

We have implemented a web-based application in order to control the RISCbot II. The framework is implemented with ASP .NET 2.0 in *C#*. The Web Form communicates with the physical robot using Web Services, which are also implemented in *C#*. In this application, a graphical user interface is provided to RISCbot's teleoperator to allow him to interact with the robot. There are two modes to control the robot; autonomous mode and manual mode. Since the RISCbot II is able to carry out a number of tasks, this web-base console allows us to hookup the video/sensor data from the robot with various computer vision applications. The architecture of the web-based application for RISCbot II is shown in Figure 22, the home page is shown in Figure 23 and the manual control mode is shown in Figure 24.

5.5 Manipulability

Studying the performance characteristics of the robot, such as dexterity, manipulability, and accuracy, is very important to the design and analysis of a robot manipulator. The manipulability is the ability to move in arbitrary directions while the accuracy is a measure of how close the manipulator can return to a previously taught point. The manipulability index is considered as a quantitative and performance measure of the ability to perform some tasks. This measure should be taken into consideration in the design phase of a serial robot and also in the design of control algorithms.

In [13], we presented a new method for measuring the manipulability index, then justify this concept by visualizing the bands of this index resulting from our experiments implemented on different manipulators such as the Puma 560 manipulator, a six DOF manipulator and the Mitsubishi Movemaster manipulator.

Furthermore, we use the proposed method for measuring the manipulability index in serial manipulators to generalize the standard definition of the manipulability index in the case of mobile manipulators. The proposed algorithm is described in details in [13].

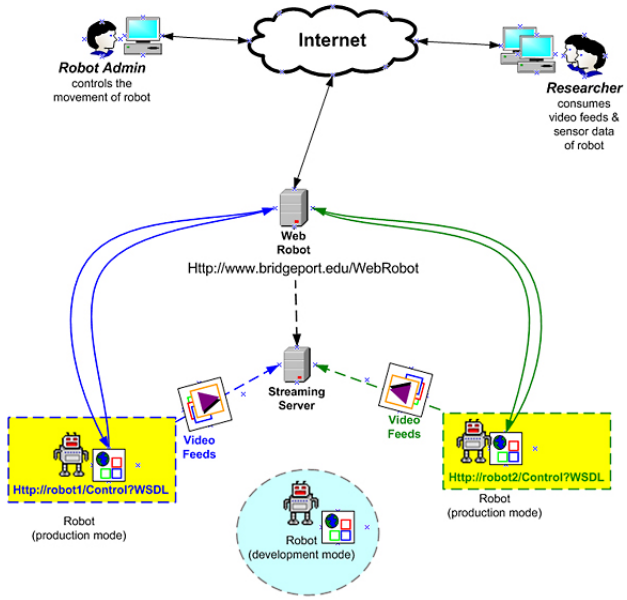

Fig. 22 RISCbot II website Architecture.

Fig. 23 RISCbot II Website.

5.6 Navigation and Obstacle Avoidance

A prerequisite task for an autonomous mobile robot is the ability to detect and avoid obstacles, given real-time sensor readings. Given partial knowledge about the robot's environment and a goal position or a series of positions, navigation encompasses the

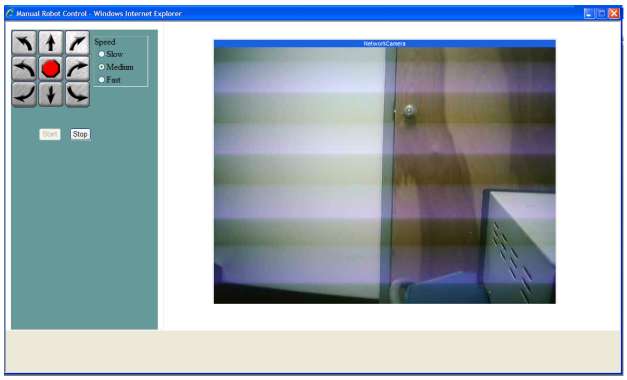

Fig. 24 web-based manual control.

ability of the robot to act, based on its knowledge and sensor values so as to reach its goal positions as efficiently and reliably as possible. The techniques used in the detection of obstacles may vary according to the nature of the obstacle. The resulting robot motion is a function of both the robot's sensor readings and its goal position. The obstacle avoidance application focuses on changing the robot's trajectory, as informed by sensors during robot motion. The obstacle avoidance algorithm, which is used in RISCbot II, is the Artificial Potential Fields Algorithm [14].

6 Implementation and Results

The RISCbot II platform differs from any related robotic platforms because its mobile platform is a wheelchair base. Thus, the RISCbot II wheelchair base offers many advantages: the ability to support a high payload, high speed motor package (the top speed of the wheelchair is 6 mph), Active-Trac and rear caster suspensions for outstanding outdoor performance, and adjustable front anti-tips to meet terrain challenges.

In order to use the wheelchair as a mobile platform , a reverse engineering process is used to understand the communication between the wheelchair's joystick and the motor controller. This process is performed by intercepting the continuous stream of voltages generated by the joystick after opening the joystick module and reading the signals within joystick wires that are sent the signals to the wheelchair controller.

We use different types of sensors so that the RISCbot II can perceive its environment with better accuracy. Our robot hosts an array of 13 $LV-MaxSonar^{®}-EZ0^{TM}$ ultrasonic sensors[15]. The working envelope of the 13 sonars is shown in Figure 25. The sensors are suitable for obstacle avoidance applications but their wide beams are unable to distinguish features within the beam angle, making sonars a poor choice of sensor for fine feature extraction within indoor environments. This resolution problem is magnified for objects further away from the robot (i.e., objects appearing at the wide end of the beam). Lastly, our robot is also equipped with an array of 11 Sharp GP20A21YK infrared proximity sensors above the sonar ring. The sonar and infrared sensors are mounted together so that their beams are oriented in the same direction. The configuration of sonar and infrared sensors is shown in Figure 26. These sensors allow the RISCbot II to obtain a set of observations and provide these observations to the controller and higher decision making mechanisms. The controller acts upon this

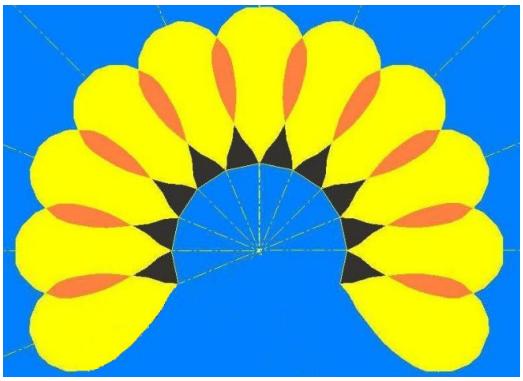

Fig. 25 Working Envelope of EZ0 sonar Sensors.

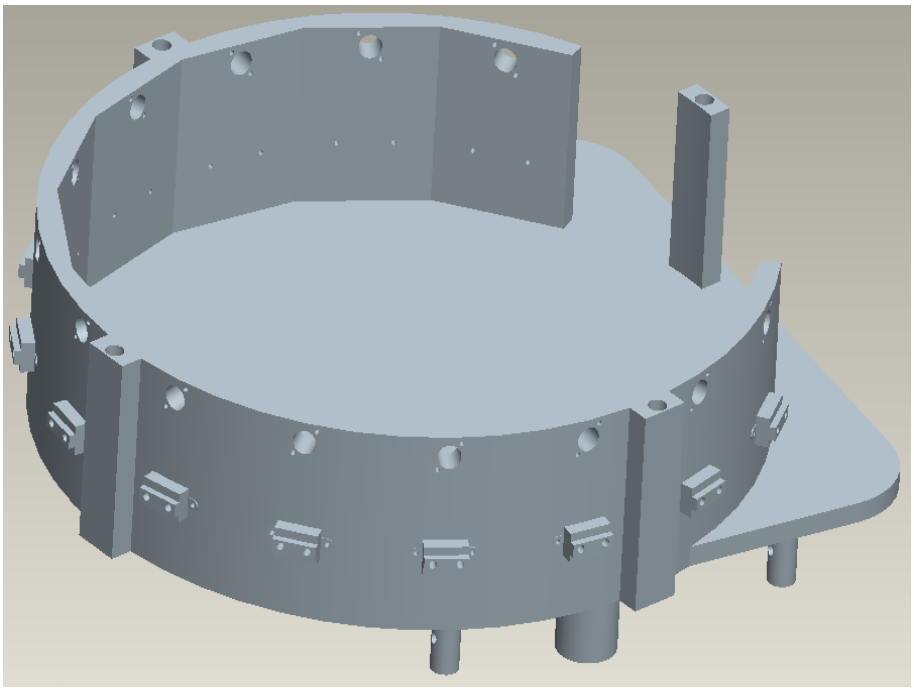

Fig. 26 A closeup view of the sonar and infrared sensors array.

set of observations to cause the robot to turn in the correct direction. The integration of these modules together constitutes an intelligent mobile robot.

A main drawback of these sensors is that they can only accurately measure obstacle distances within a range of 0.1m to 0.8 m. Another drawback of these sensors is that they are susceptible to inaccuracies due to outdoor light interference as well as an obstacle's color or reflectivity characteristics, which can be seriously affected by windows and metallic surfaces.

Note that since our sonar and infrared sensors are in fixed positions, our experiments concentrated on performing data fusion on data obtained from a particular

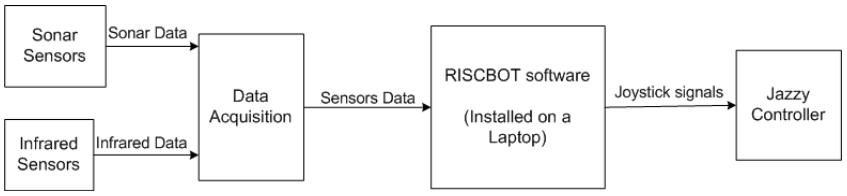

Fig. 27 The components of the RISCbot II system.

fixed height in the environment. In this project, sonar and infrared sensors are used together in a complementary fashion, where the advantages of one compensate for the disadvantages of the other.

As shown in Figure 27, the RISCbot II software, which is written in Visual *C#* and runs on a laptop reads the values of all sensors at a rate of 10 HZ gathered in data acquisition.

The RISCbot II software maps the sensory inputs to a series of actions used to achieve the required task. Based on the used algorithm, the RISCbot II software responds to the sensor data and generates a corresponding stream of voltages to the wheelchair controller. These voltages control the direction and the speed of the wheelchair to cause the RISCbot II to turn in the desired direction.

The experimental result indicates that the RISCbot II can detect any unknown obstacle and avoid collisions while simultaneously steering from the initial position toward the target position.

7 Conclusions and Future Work

In this paper, the mobile manipulation platform RISCbot II has been presented. The RISCbot II platform differs from any other robotic platform because its mobile platform is a wheelchair base. Thus, the RISCbot II has the advantages of the wheelchair. Furthermore, the RISCbot II consists of a comprehensive sensor suite, and significant end-effector capabilities for manipulation. In addition, we have used infrared and sonar sensors to monitor any type of obstruction in the robot's path. This research aspires to find real-time collision-free trajectories for mobile manipulation platforms in an unknown static or dynamic environment containing some obstacles, between a start and a goal configuration. Path planning for mobile robots is one of the key issues in robotics research that helps a mobile robot find a collision-free path from the beginning to the target position in the presence of obstacles. Furthermore, it deals with the uncertainties in sensor data.

The objective for this project is to implement a general purpose mobile manipulator that can be used in multiple applications, such as teleoperation, navigation, obstacle avoidance, manipulation, 3-D reconstruction, map building, face detection, and face recognition. There are great benefits for using a mobile manipulator in dangerous, inaccessible and toxic environments. In teleoperation, a human operator controls the RISCbot II from a distance. The teleoperator has some type of display and control mechanisms, and the RISCbot II has sensors which gather all the information about the remote environment, an end-effector, and mobility.

In our anticipated future work, there will be an ongoing effort to develop multiple mobile manipulation systems and platforms which interact with each other to perform

more complex tasks exhibiting intelligent behaviors utilizing the proposed manipulability measure.

References

1. K. Konolige, K. Myers, E. Ruspini, and A. Saffiotti, "The saphira architecture: A design for autonomy," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, pp. 215–235, 1997.
2. A. H. Indratmo, "A tool for mobile robotics research," *Jurnal Teknik Elektro*, vol. 2, no. 1, 3 2002. [Online]. Available: http://puslit.petra.ac.id/journals/articles.php?PublishedID=ELK02020103
3. M. Lindstrom, A. Oreback, and H. Christensen, "Berra: a research architecture for service robots," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 4, 2000, pp. 3278 –3283 vol.4.
4. R. Volpe, I. Nesnas, T. Estlin, D. Mutz, R. Petras, and H. Das, "The claraty architecture for robotic autonomy," in *Aerospace Conference, 2001, IEEE Proceedings.*, vol. 1, 2001, pp. 1/121 –1/132 vol.1.
5. I. A. Nesnas, R. Simmons, D. Gaines, C. Kunz, A. D.-C. T. Estlin, R. Madison, J. Guineau, M. McHenry, I.-H. Shu, and D. Apfelbaum, "Claraty: Challenges and steps toward reusable robotic software," *International Journal of Advanced Robotic Systems*, vol. 3, no. 1, pp. 023–030, 2006.
6. I. A. D. Nesnas, A. Wright, M. Bajracharya, R. Simmons, and T. Estlin, "Claraty and challenges of developing interoperable robotic software," in *In invited to International Conference on Intelligent Robots and Systems (IROS*, 2003, pp. 2428–2435.
7. S. Bergbreiter and K. Pister, "Cotsbots: an off-the-shelf platform for distributed robotics," in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 2, 27-31 2003, pp. 1632 – 1637 vol.2.
8. "Real time obstacle avoidance for a riscbot ii with a navigation task," online: http://www.youtube.com/watch?v=EmQxAS0TNn0, April 2009.
9. A.-E. Hassanien, J. H. Abawajy, A. Abraham, and H. Hagras, Eds., *Pervasive Computing: Innovations in Intelligent Multimedia and Applications.* New York: Springer, 2009.
10. R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, 1st ed., ser. Intelligent Robotics and Autonomous Agents series. The MIT Press, April 2004.
11. P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR 2001*, vol. 1, 2001, pp. I–511–I–518.
12. R. Tjahyadi, W. Liu, S. An, and S. Venkatesh, "Face recognition via the overlapping energy histogram," in *IJCAI*, 2007. [Online]. Available: http://www.ijcai.org/papers07/Papers/IJCAI07-465.pdf
13. A. Y. Elkady, M. Mohammed, and T. Sobh, "A new algorithm for measuring and optimizing the manipulability index," *Journal of Intelligent and Robotic Systems*, pp. 75–86, 2009.
14. O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotic Research*, vol. 5, no. 1, pp. 90–98, 1986. [Online]. Available: http://portal.acm.org/citation.cfm?id=6812
15. "Lv-maxsonar® – $ez0^{TM}$ data sheet," Online: www.maxbotix.com, July 2007.