Fault detection using virtual environment and wireless robot

Emil Voisan, Constantin Volosencu, Adrian Leu, Florin Dragan
Department of Automation and Applied Informatics
University Politehnica from Timisoara
Vasile Parvan, No. 2
Romania

Abstract: - Intelligent robots that are able to efficiently interact with working environment is an actual trend today, the classic approach is to use robot sensors to discover surroundings and based on that readings to take decisions. Our proposal is based on a virtual environment to describe the objects from the real environment and based on this environment the decisions are taken and transmitted to the "real" robot.

Key-Words: - virtual, environment, robot, detection.

1 Introduction

1.1 Application overview

To study and develop robotic devices able to efficiently interact with the working environment is necessary to have a playground, a virtual room, to be able to implement and test robot behavior before its release on the real world, also this virtual environment can be used to generate decisions for the real robot when this is sensor less or sensors are malfunctioning. To obtain a realistic behavior for the real robot is necessary, for the virtual room, to realistically represent the working environment.

In this case, fault detection is based on the image processing of pictures [5] taken for each defined target from a specified area. In our specific application modeled area is a part from our laboratory and inspected surfaces are sides of desks form this location. Faulty area is marked by a white page post it on that side. Main objective is to periodically inspect specified targets and to monitor them for any modifications.

Technologies involved in this project are falling in several categories robotic devices: Dr. Robot X80, virtual environment is created using Coin3d and image processing techniques. Based on those technologies a part from laboratory is reconstructed into the virtual environment and based on this environment virtual a real robot must visit each object periodically.

1.2 Related work

There are several projects related to this project. We present short connection with following:

- Playerstage [6] project – open source project aimed to provide a common interface to various robots from application point of view, also offers possibility to simulate a robot population behavior in 2D(Stage) or 3D(Gazebo) virtual environment.
- Evaluation of Inhabitant's Walking Habit in Intelligent Space [1] is more related to this project and is oriented on helping people with disabilities, main purpose being development of a model for interaction between intelligent space and moving object that enters into simulated area.

2 Technologies

There are two main technologies involved in this project, one connected with the robot itself, X80 WiRobot, as a physical device and second is Coin3D, related with development o the virtual environment.

2.1 X80 WiRobot

WiRobot is an integrated hardware and software robotic system developed by Dr Robot, and includes the mechanical structure, electronic modules and software development kit.

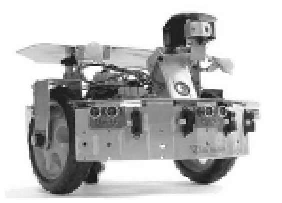
Fig 1 WiRobot X80

2.1.1 Hardware overview

The mechanical structure is pre-built with electronic components such as multimedia controller, motion sensing controller and various peripherals. Also it has a wireless module responsible with communication with a pc-based software component.

Mechanical and control components:

- Two 12V motors;
- 17,8 cm driving wheel;
- Dimensions: 38 cm diameter, 25,5 height;
- Weight: 3,5 kg;
- Fine speed and position control based on two 1200 count per wheel-cycle quadrature encoders;

Electronic system components:

- Fully integrated WiFi system supporting both UDP and TCP/IP protocol;
- Full color video and two-way audio capability;
- Collision detection sensors: 3 sonar range sensors and 7 IR range sensors;
- Two pyroelectric sensors for human motion detection.
- Temperature, acceleration/tilting sensors.

2.1.2 SDK overview

It is a PC-based software framework for robotic system development, containing facilities for memory management, system communication and user interface and utilities for audio, video, sensor data acquisition and motion control.

Programs developed using this SDK runs on PC sending and receiving data to a WiRrobot via a wireless connection. The firmware on the embedded controller takes care of low level operations: motion control, audio-video capture and compression, audio playback and wireless communication.

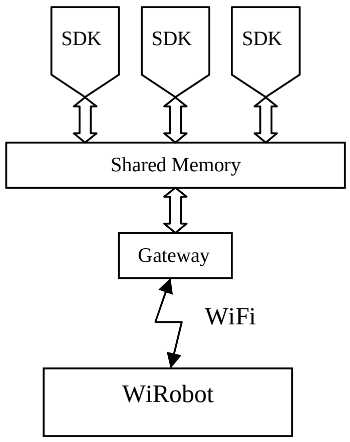

Fig 2. Software Architecture

User programs use a SDK component to communicate with WiRobot controller. All data transfers between user program and WiRobot are managed by a Gateway program which communicates with SDK component on shared memory.

2.2 Coin3D

Is a library set used to create 3D graphic applications. Portable over a wide array of operating systems: UNIX/Linux, Windows and Mac OS X and compatible with SGI Open Inventor [2].

Main components:

- Coin – C++ API for 3D graphical libraries, is a abstract level over OpenGL with many complex optimization, transparent for programmer.
- Graphical Interfaces – libraries to interface Coin with GUI on host operating system: Microsoft GUI on Windows or Qt, Xt on XWindow[4].
- Import/Export libraries – for managing various file formats.

Coin3D is based on OpenGL and offers a variety of graphical objects that can be used, modified or extended according to application.

Objects form this library includes database primitives such as: objects of the rendering engine, shape, properties and group nodes, also interactive manipulators and components material editor, light editor and visualization windows. Is based on an object orientated system and offers capabilities to transfer data between applications based on a specific file format.

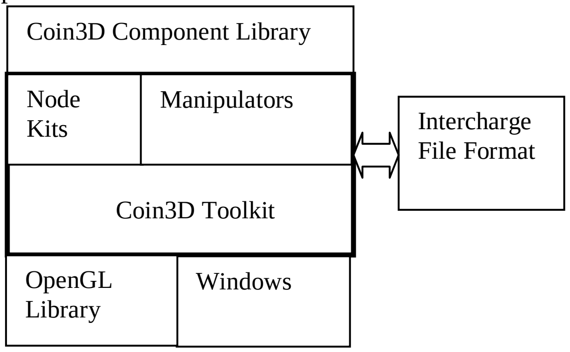

Fig. 3 Coin3D Structure

Based on OpenGL, Coin3D offers a programming model and a interface to OpenGL. Is independent to the graphical interface system, a library, specific to interface, will bound them together.

Coin3D main focus is creation of 3D objects, all information regarding those objects, shape, size, color, texture, placing in 3D space is stores into a scene database, this database being used to generate a 3D image from existing information.

Coin3d is a high level programming language that uses OpenGl routines to create 3D content. Objects defined in Coin3D encapsulate data regarding display, reading, writing and other state information. Each object can be subjected to various operations such as: selection, search in database, computation of its bounding box. Such an object will be displayed when its associated display method is called.

Coin3D facilities:

- Scene database **-** includes: properties, node kit, sensors. It is use to create a hierarchical 3D scene.
- Node kits – mechanism to group nodes.
- Manipulators – objects from database available for user interaction.
- Components – render window library (specific to particular windows system: Xt, Qt, Microsoft Windows), material editor, display tools, etc.

3 Fault detection – application set-up

In case of this application fault detection presumes patrolling a designated area and inspecting general appearance of target objects. For this particular case the working environment is a part of our laboratory (Fig. 5) and idea of fault, means changing of appearance for target objects.

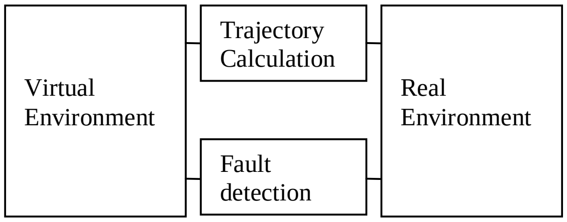

Fig. 4 Application structure

From application point of view main application parts are Virtual Environment, Trajectory calculation, Fault detection and robot as part of the real environment (Fig. 4).

3.1 Virtual Environment

Recreates the real environment into which robot will operate. As a restriction, it must be accurate because all decisions transmitted to robot are based on the information acquired from this.

Virtual environment contains several objects: chairs, desks, book shelf. Visual recreation of those objects is done with the help of Coin3D, for each object a specific class to describe general characteristics has been developed. Particular parameters for each object, including position are stored into a database. New objects can be inserted into virtual environment only before application is launched.

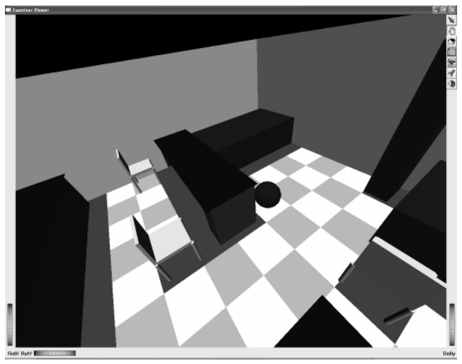

Fig 5. Virtual Room

Each object is constructed from basic shapes like Cube, Sphere, and Cylinder parameterized and translated accordingly to recreate specific part of the object. All those basic shapes are graphical primitives offered by Coin3D toolkit.

Floor pattern, has no correspondence into the real environment and is used for robot positioning and trajectory calculation.

One specific problem is robot movement into virtual environment. In this case is based on a asynchronous event, alarm, that for each occurrence will translate the object with a small amount until it reaches its intermediate/final destination.

3.2 Real Environment

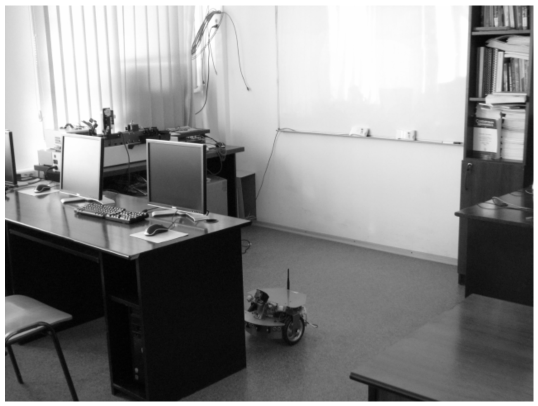

Fig 6. Real Environment

As presented, working environment is part of our laboratory (Fig 6), and recreates only objects that can interfere with robot path.

It is important to ensure that the real environment and the virtual one are synchronized. This involves two aspects: keeping the same proportions in object construction into the virtual environment as objects in real world and using the same starting point for both virtual and real object.

2.1.1 WiRobot

It is relatively easy to operate such a robot and application is not requiring special capabilities for robot. From the movement point of view there are implemented a series of routines that permits to move the robot to move robot forward or backward for a specified distance, or to rotate with specified degrees (in this case 90). From the picture capabilities point of view, based on the SDK, it is possible to take a picture and to transfer to the base workstation where all computations are done.

3.3 Trajectory calculation

Robot trajectory consists on several check points that must be reached by robot in order to do fault tests. Robot movement is achieved in discrete steps of 50 cm (selected to match the floor pattern).

Robot has the freedom to do four types of movement: forward, backward, left and right. For each step robot chooses direction to move based on following algorithm: For each possible direction calculates the distance between coordinates of the resulting position and the coordinates of the target position Fig. 7. The robot will move along available direction that provides shortest distance to the target position.

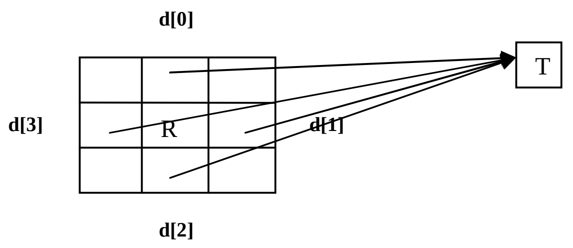

Fig. 7 Robot direction

To establish if a direction is available, is used a matrix that encodes with values of 0 or 1 status of each box from the floor. In the virtual representation of the working environment unavailable zones are marked with red Fig. 5.

3.4 Fault detection

Detectable faults are color modification for target surfaces. For each picture taken an arithmetic mean is calculated based on the RGB value for each pixel.

In order to obtain the reference values for each target, before entering into fault detection mode, robot must perform a calibration. Into the calibration mode robot will visit each target node and take a initial picture which stands as reference for comparison when robot runs in fault detection mode.

It is considered that in the calibration mode none of the target objects is faulty.

4 Conclusion

Proposed solution has the advantage that can use relatively cheap robots that require only movement capability and possibility to take pictures. Also computational power of the robot is irrelevant, main decisions are taken by the on the platform that runs the virtual environment, in this case a regular PC with Windows XP. In fact all decisions are moved to that platform, rendering the robot to a simple participant to the environment.

An important disadvantage for this solution is the fact that it is possible to loose synchronization between virtual robot and real robot, fact that requires an external factor to place the robot into a known position.

As future development presence of sensors capable to detect if new objects walk into the working environment and to reflect that into the virtual environment will greatly improve reliability of such a system. Also based on such sensors it is possible to resynchronize real and virtual robot without any external intervention.

References:

[1] P. Szemes, H. Hashimoto, E. Voisan, F. Dragan *Evaluation of Inhabitant’s Walking Habit in Intelligent Space*. IECON, 2003. pp.1390-1395, 2003.11, Roanoke, Virginia

[2] J. Wernecke, *The Inventor Mentor Programming Object-Oriented 3D Graphics with Open Inventor*. Addison-Wesley, 1994.

[3] J. Friel, *Practical Guide to Image Analysis*, ASM-Intl. 2000.

[4] J. Wernecke, *The Inventor Toolmaker*, Addison-Wesley, 1994.

[5] J. Bronstein, Y. Koren, Real-time obstacle avoidance for fast mobile robots., IEEE Transactions on Systems Man and Cybernetics vol. 19, no. 5, pp 1179-1187.

[6] B. Gerkey, R. Vaughan, A. Howard. *The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems*. ICAR 2003, pages 317-323, Coimbra, Portugal, June 2003.