Hardware-software Solutions to Surveillance an Objective

Emil Voisan, Daniel Iercan, Florin Dragan, Onut Lungu

University “Politehnica” from Timisoara, Faculty of Automatics and Computer Science, V. Parvan no. 2, Timisoara, emil.voisan@aut.upt.ro

Abstract: This paper shows the automatic supervision possibility of different objectives using a camera and a computer/laptop. For connecting the camera to the computer and for command it, it was realised a hardware-software interface. Using this interface we can rotate the camera using a small engine so we can have a good view of entire objective. The user program is developed using Kylix and permits an exact control of the camera moving in operator mode and in automatic mode. We can also introduce pre-established positions of the camera. Virtual representation of surveilled area is also available to determine best locations to place surveillance hardware.

Keywords: hardware interface, kylix, coin, virtual

1 Introduction

Different objectives need the presence of a human observer 24 hours from 24 hours. For avoiding this inconvenient for each objective we could use a camera and centralize data on a PC/laptop. An important issue in this case is the mobility of the camera regarding the overseen objective.

The general scheme of this surveillance system is presented in Figure 1.

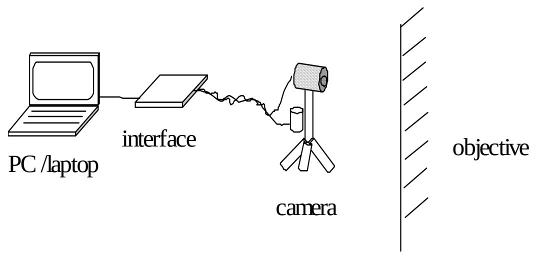

Figure 1
General scheme of the surveillance system

This system is realized using a microcamera connected at one computer or laptop and controlled on the movie plan by a rotative stand. The computer or laptop can be remote accessed by Internet or intranet and permits to connect few such systems on a single display.

The novelty of this system consists on the operating system – Linux – and the simplicity of the hardware interface.

The camera is connected at the PC/laptop on the parallel or USB port. The image acquired using this camera can be stocked on the HDD of the PC/laptop and prepared with filters or different compression algorithms depending of the requirements. On the other hand, the system can be modified and the camera gain new degrees of freedom in the sense that it's possible to control the movie using more motors.

2 Hardware Interface Description

The implementation of this surveillance solution implies hardware and software aspects. On the hardware side it's very important an easy design for the positioning of the camera. On the software side we must take in consideration the control of the hardware designed and the integration of the images flow on the application program realized.

On the hardware design of the interface between PC/laptop and the rotative stand we must take in account some simple and efficient solutions for a performance/costs good rate. So, the orientation of the camera must be very precisely in horizontal and vertical plan to take a good panorama of the entire objective. The positioning of the camera must have a low speed to avoid the smoke effects and the space of rotation must be very clear to avoid the break of the connection cables.

On the horizontal plan, the rotation axe pass on the vertical maintenance arm of the camera. On the vertical plan, the rotation axe is perpendicular on the first axe. Taking in account these aspects, the stand need two small motors, one for each axe.

Because the camera isn't heavy (about 100 g), the maintenance arm is fixed on the basis with a bearing and it's from aluminium, the solution for control the stand is two step by step motors of small powers which are able to a precise orientation. The used motors are STH-39D103-01 models, made in Japan and with the following characteristics:

- an angle of 1.8 degrees/step;
- 0.16 absorbed current;

- 4 phases, with 6 conductors;
- 12 V tension.

In the simple version, when the distance between computer and the camera is about 5-6 m, the interface between the computer and the two motors can be realized on the principle of using transistors with power switch role. But in this case it’s obvious that the needing for the control mean 8 transistors and it’s more simple to use a driver ULN200 with 7 lines of command/outputs, an integrated monolithic specialised circuit. This circuit permits a simultaneous control of 7 lines of medium power (0.5 A) at a tension of max 50 V. Taking in account the characteristics of our motors it’s clear that this circuit satisfy our needs for control. For controlling the 8 wrappings of motors we use one line to command two wrappings of different motors.

An other problem solved was the limitations of the moves on the horizontal and vertical axes to avoid the break of the cables or the detaching of the camera. For this scope the stand was foreseen with 2 optoelectronics sensors. The scheme for the hardware interface between computer and camera is shown on the appendix.

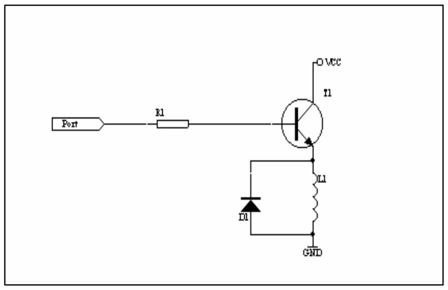

Figure 2
Simple control for one motor

3 Virtual Environment

COIN3D is a high level library built over OpenGL running over a wide range of Operating Systems - Unix based, Windows and MacOS. To generate 3D scenes makes use of an scene graph, development speed and easiness increasing versus OpenGL. Also COIN3D is fully compatible with well known Open Inventor. Extra features being support for VRML, 3D sound, 3D textures and parallel processing. All those features made this toolkit to be the natural choice for developing the application.

COIN3D is only the 3D rendering library, all other interactions with Operating System (window operations, handle user input) must be accessed through other libraries like SoWin(Windows), SoQT (Linux, Windows), SoXt(Linux). Library of choice for this project was SoQt for its versatitlity and cross platform spread.

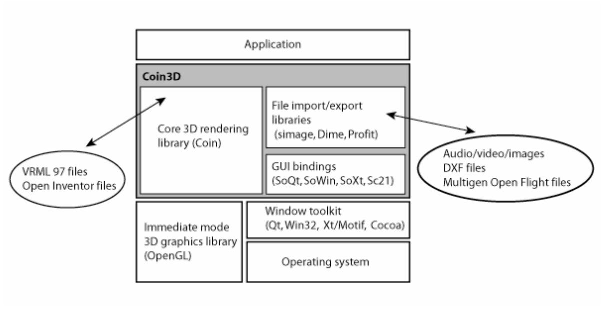
Application
Coin3D
Core 3D rendering library (Coin)
File import/export libraries (simage, Dime, Profit)
GUI bindings (SoQt, SoWin, SoXt, Sc21)
VRML 97 files Open Inventor files
Audio/video/images DXF files Multigen Open Flight files
Immediate mode 3D graphics library (OpenGL)
Window toolkit (Qt, Win32, Xt/Motif, Cocoa)
Operating system

Figure 3
Coin 3D structure

Foundation in 3D scene rendering is the node object, it has associated properties such: object shape, material properties, geometric transformations, etc., properties that also can be nodes, basically we will end with a tree like structure representing scene database.

Important elements to create a scene are:

- scene objects – objects that will be rendered on the screen
- lights – necessary to iluminate the scene.
- cameras – needed for scene visualization.

To describe an object into the Open Inventor terminology the object must be disassembled into its parts and each of it needs to be specified separately – Figure 4.

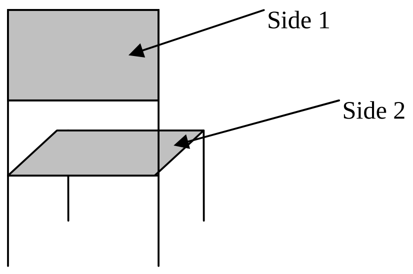

Figure 4
Chair component parts

As it can be seen in this figure the chair can be divided in three parts, each part being derived from a cube class. Each part can be implemented using the following components:

- separator: to isolate the side specific properties.
- shape: specific shape of the object – in this case cubic.
- translation: will translate the object to its custom position – according to the object specifications.

This structure can be represented through a graph like in Figure 5:

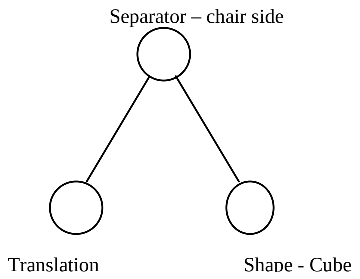

Figure 5
Graph components

In order to create the entire scene graph the only operation that needs to be performed it is to put the graph for each component into a more complex structure.

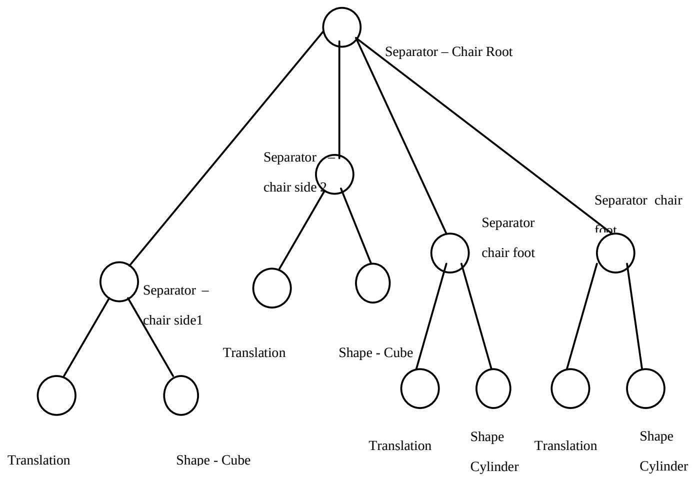

Figure 6
Desk Scene Graph

As it can be seen the complexity of the scene graph grows with the number of the objects included into the final simulation.

To apply an action to the scene graph (such rendering) first it is needed an instance of the action class and second it will be applied to the root node of the scene graph. Typically, executing an action involves traversing the graph from top to bottom and left to right. During this traversal, nodes can modify the traversal state, depending on their particular behavior for that action.

Another important aspect that concern the placement of the objects into the virtual space it is the coordinate system which is a right handed one, with z axe coming out from the screen. The objects are described in their own local coordinate space, and only after the transformations have been applied the object it is integrated into the world coordinate space.

4 Software Interface

The software module of the application was developed on Linux using the Kylix software development suite and allows a precis control of the camera movement.

The software application must cover two aspects – the control of the hardware interface and the integration of the video stream. The hardware interface is controlled through the parallel port of the computer and the image it is acquired scanning the USB port where the WebCam is connected.

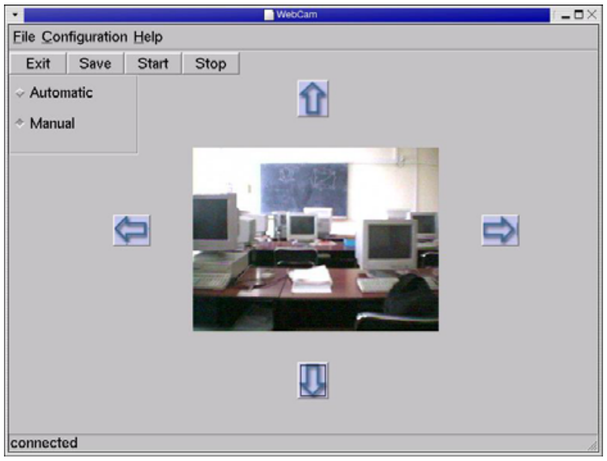

Figure 7
Software interface

Concerning the camera movement control we considered that the application must have two modes of work – automatic and manual. Within the manual mode the user can change camera orientation based vertical and horizontal planes. Restrictions that can appear are concerning the speed of the camera rotation, a too high speed can determine “blackouts” on the acquired image.

From the video acquisition point of view the software module is able to display the video stream and if it is necessary to save the images on disk, using different compression algorithms.

Using the virtual environment to avoid large amount of data transferred on the network we simulate the same objective:

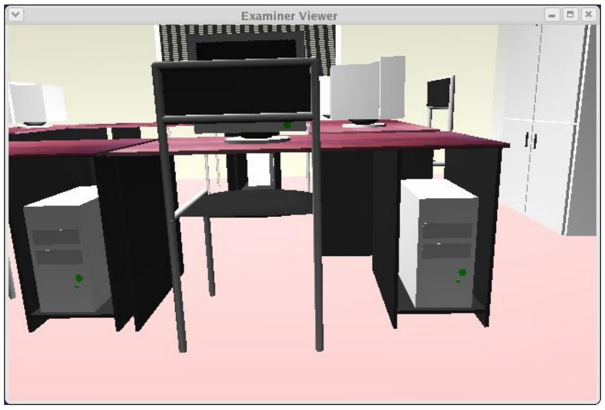

Figure 8

Virtual objective

Conclusions

The paper shows in fact 2 modes for surveillance an objective. One based on a camera postioning control and one with a virtual point of view. The objective could be so different and the virtual software permitt us to model a large scale of objects. Using virtual surveillance it could be attractive from placing surveillance camera point of view. In a large scale environments with multiple target objects in surveillend area virtual model for this environment can be used to determine ideal position for surveillance cameras in relative simple manner.

Future improvement for this project is a connection,synchronisation, between surveillance cameras and virtual model for targeted environment. In this case the main advantage of the virtual environment is a more comprehensive overview of the entire scenery.

References

[1] Josie Wernecke, The Inventor Mentor : Programming Object-Oriented 3D Graphics with Open Inventor

[2] Jacobs, Jon Q., Delphi developer's guide to OpenGL, 1999

[3] Neil Matthew, Richard Stones, Alan Cox, Beginning Linux Programming

[4] Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman, Linux Device Drivers

[5] Open Inventor Architecture Group, “Open Inventor C++ Reference Manual”, Addison-Wesley Publishing Company, 1994

[6] Open Inventor Architecture Group, “Open Inventor: Nodes Quick Reference”, http://techpubs.sgi.com/library/, 1994

[7] P. Korondi, H. Hasimoto, “Intelligent Space, As An Integrated Intelligent System”, Electrical Drives and Power Electronics, Slovakia, 2003

Appendix

Hardware interface