Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Aided Automation
Vienna University of Technology
Favoritenstr. 9/183-2
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18351
Fax: +43 (1) 58801-18392
E-mail: hassan@dbai.tuwien.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-119 May 13, 2008

A Demonstration for the Smart Room[1]

Tamir Hassan, Walter G. Kropatsch, Adrian Ion

Abstract

This report describes the work carried out between January and April 2008 to create a demonstration for the Smart Room which involves tracking a blind (or blindfolded) person from several cameras in the room, which contains an obstacle. Feedback is generated by means of audio tones from four loudspeakers positioned in the periphery. This report describes in detail how the set-up can be recreated, and suggestions are also included for continuation work.

[1]This work is partially supported by the Austrian Science Fund under grant S9103-N13.

Contents

1 Introduction

In the months prior to this work, PRIP invested in the hardware for a Smart Room; a large room with a number of video cameras and related equipment, enabling the practical development and trialling of intelligent solutions based on video acquisition. This report describes the work undertaken between January and April 2008 to create a demonstration for this newly acquired equipment. The purpose of such a demonstration is two-fold: First, it would help to justify the expenditure by demonstrating the equipment's use in solving a practical problem. Secondly, such a demonstration, when carried out e.g. at the Beginners' Day, could attract potential students to the PRIP Department, and give them an insight into the research work that is being carried out.

The demonstration described in this report involves tracking a person from several cameras around the periphery of the room, which contains an obstacle. It is assumed that this person cannot see (e.g. he is blindfolded). Feedback is given by means of audio tones generated from four loudspeakers, which help the person avoid the obstacle.

The emphasis of this work was not to attempt to recreate a full 3D model of the scene and consequently obtain precise positioning information of the objects within, but rather to obtain simple relationships, from which an approximate position can be deduced and feedback can be given.

This report describes the proposed demonstration in detail, how the set-up can be recreated, and suggestions are also included for continuation work.

2 Physical setup

2.1 Cameras

The room is surveilled by three cameras, located in a triangular form, as shown in Figure 1.

Cameras C0 and C1 were placed on stands (actually lamp stands) set to a height of approximagely 1.5 m. Using these stands, it was not possible to tilt the cameras up or down; the cameras were level.

Camera C2 was placed on a tripod at a height of approximately 1.8 m and tilted approximately 20 deg downwards.

Camera C2 has a fixed focal length. Cameras C0 and C1 are fitted with zoom lenses; C1 was set to maximum wide angle, whereas C0 was set approximately 2/3 towards maximum wide (1/3 tele). All cameras were focussed to their apparent hyperfocal distances (so that the walls of the room were

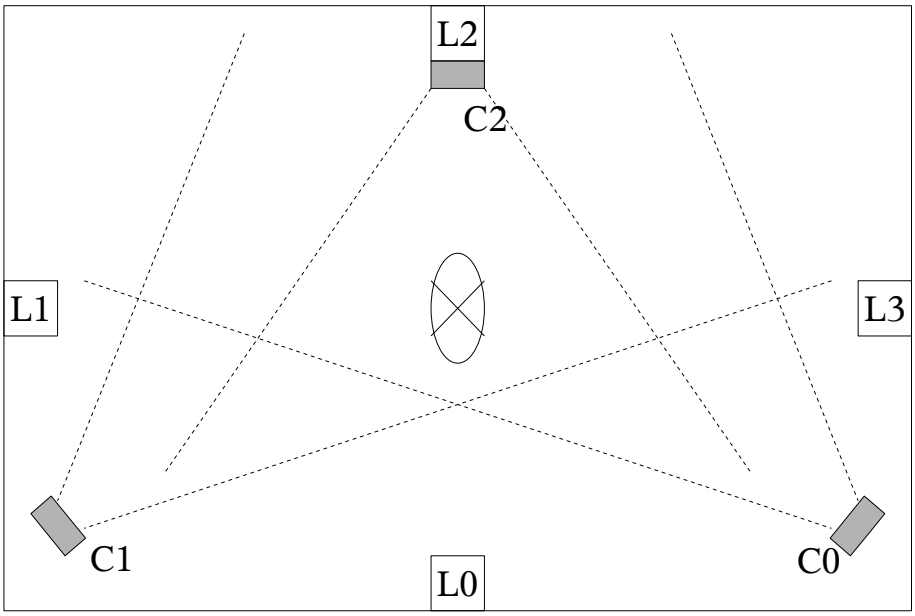

Figure 1: Room layout

rendered with maximum sharpness).

All three cameras are IP cameras, which were connected to a switch and configured to the subnet 192.168.10/8, with the following properties:

Symbol	Name	Model number	IP address	MAC address
C0	East camera	Axis 211	192.168.10.2	00408C6F9288
C1	West camera	Axis 211	192.168.10.3	00408C6F9418
C2	North camera	Axis 2100	192.168.10.4	00408C5928D1

The computer used is the CABS workstation with the IP address 192.168.10.1. This was entered as the *alternate configuration*, to allow switching between the institute (PRIP) and local networks without reconfiguration.

If this set-up is to be recreated at a later date, it might be necessary to reassign the IP addresses using e.g. the following command (entered into the Windows shell): `arp -s 192.168.10.2 00-40-8C-6F-92-88`

Alternatively, the *Axis IP utility* for IP address assignment may be used: `http://www.axis.com/techsup/software/iputility/index.htm`

In order to verify successful IP address assignment, the command `ping` may be used.

Each of the cameras has a web interface for set-up and preview. For camera C2, in order to receive video at the correct resolution, this needs to be set to 320x240 within the web interface. For the other cameras, the resolution is passed as a parameter during frame-grabbing, and the resolution setting in the web menu has therefore no effect on the program.

For all cameras, compression, colour and exposure were set to medium in the web interface. Image type was set to single images (JPEG) and not video (MJPEG). This seting also only effects camera C2; the image format (MJPEG or JPEG) on the other cameras is determined automatically from the file extension in the HTTP request.

2.2 Loudspeakers

The Vivanco quadrophonic sound card was installed in the CABS workstation. The loudspeakers L0-L1 and L2-L3 are two pairs of inexpensive stereo computer speakers, whose interconnecting cables have been extended to allow free placement within the room. Because the room has four walls, it was decided that the most intuitive form of feedback would be sounds coming from each of the directions of the walls. Therefore, the speakers were placed in the centres of the arena walls and not the corners.

The loudspeaker pair L0-L1 is connected to the front output of the sound card, whereas the pair L2-L3 is connected to the rear output. The volume

control on the host PC is set to maximum; the volume control on each pair of speakers was adjusted so that front and rear are equally loud.

2.3 Computer system

The system is driven by the CABS workstation (Pentium IV 2.8 GHz, 1 GB RAM, operating system: Windows XP), and the code has been developed in Visual C++ using Visual Studio 2003. To perform image manipulation, the *Open CV Library* is used. In order to play 3D sound, *DirectX 9.0* and its SDK have been installed.

3 How the system works

Currently, the images are obtained over HTTP using the `WinINet` library functions. This method was chosen after consulting with the OpenCV discussion group (`http://tech.groups.yahoo.com/group/OpenCV`).

The function `CVCaptureFromCam` performs the entire process of interfacing with a webcam and allowing the individual frames to be addressed. Unfortuately, this function does not work at all with IP cameras, and it was therefore necessary to re-implement the complete procedure.

During investigation of the possible implementation methods, the page `http://www.enib.fr/~veyret/links.html` was found, which contains two libraries, `axis` and `network`, which together claim to provide access to the images generated by Axis IP cameras. However, after several attempts, it was deemed to complex to compile and integrate these libraries into the code, and an alternative solution was sought.

The procedure of obtaining the images from HTTP is as follows:

- First, the URL is build using the camera number
- `InternetOpen` is called to open a handle for an HTTP request
- `InternetOpenURL` is called to perform the request and create an array of file objects `hFile[c]`
- From the `hFile` objects, the data is read in 1 k chunks into a buffer `buffer`
- An `IPicture` object `gpPicture` is created to load this image data using the function `OleLoadPicture`
- A blank `HBitmap` `hbmpTemp` is created, with the same pixel dimensions as the image

- `gpPicture` is then rendered onto bitmap `hbmpTemp` (using the method in `http://www.joachimrohde.com/cms/xoops/modules/articles/article.php?id=51`
- The function `hBitmap2Ipl(hbmpTemp)` is called, which converts `hbmpTemp` into an `IplImage` (the image format used by OpenCV) (using the method in `http://rocee.bokee.com/2409831.html`)
- The resulting image is flipped vertically

This method achieves a frame rate of between 2 and 4 frames per second, depending on resolution and the number of cameras which are in use simultaneously (2 fps with 3 cameras running at 320x240). A higher frame rate would be desirable, and one of the suggestions for future improvement is therefore to investigate alternative approaches to obtaining data from the cameras with a view to increasing the frame rate that can be obtained.

Please note that this method does not currently work with the MJPEG format. Streaming MJPEG directly from the cameras instead of JPEG could possibly provide the desired performance improvement.

3.1 Tracking

It was not the objective of this exercise to work on tracking algorithms, and therefore a simple approach based on chroma values was used. Form each of the three camera images `frame[c]`, the image is first converted into the HSV colour space (`hsv[c]`), and from then on to a binary format, `binman[c]` and `binobs[c]`, in which red and cyan pixels respectively are represented with a value of 1; all other pixels with 0.

These binary images are then eroded and dilated with a kernel of radius 2, to ensure that only significantly large red or cyan areas are detected as objects or interest.

The colours red and cyan were chosen because of their availability and their dissimilarity from each other and from background objects in the scene.

From these binary images, the maximal, minimal and mean x and y values are calculated. These are then painted and overlaid on the image. The function `findObject` is used to determine whether the binary image (after erosion and dilation) contains an object (number of white pixels is above a certain threshold) and, if an object is found, these aforementioned co-ordinate values of the object.

3.2 Output

The sound output is adapted from the DirectX demo application `Play3DSound`, which plays a wave file continuously (`DSBPLAY_LOOPING`), allowing volume and positioning to be adjusted. This sound can be played or stopped in the code.

The sound contained in the file `ding_mono.wav` is the sound currently being used. This is a mono conversion of the file `ding.wav`, which ships with Windows.

In the code, `t` represents the direction (speaker number) and `v` the volume. The volume is defined on a logarithmic scale and ranges from -20000 (silence) to 0 (maximum loudness). In this program, two loudness levels are used; -1500 and -200. As a result of an experiment carried out, we found that the ear can not accurately distinguish beween many more loudness levels than this.

It is also possible to play the sound just once, and to stop it while its playing. This could be used to control the frequency of repetition, much like parking sensors on modern cars. This could be more intuitive for the user than the current solution of two volume levels.

The ability of the user to locate a sound coming from one of the four speakers did not work as reliably as originally envisaged. It is worth noting that, even with the direction set to maximum left or right, the speakers on the opposing side did not mute completely with Direct3D. It is certainly worth investigating if the output could be improved, e.g. by using pitch or different sounds to represent direction. This is described further in the final section of this document.

Furthermore, the reader should be made aware that Direct3D is actually intended for games programming, and its strengths lie in improving the effect of realism. It is perhaps less suited for our purpose. In the initialization stage, there is a choice of three algorithms for 3D sound processing, which did not appear to have had any perceptible effect in our application.

3.3 Determining which output to sound

The series of logical if-then-else statements towards the end of the main loop (as shown in the Appendix) are used to calculate which output should sound. As there are only three cameras (and no scene model; only simple relationships are used), the position of the objects on the scene cannot always be accurately determined. Figure 2 summarizes the algorithm.

If the user is to the side of the obstacle (in the direction of the west or east camera), but is picked up in the centre (north) camera, the distance

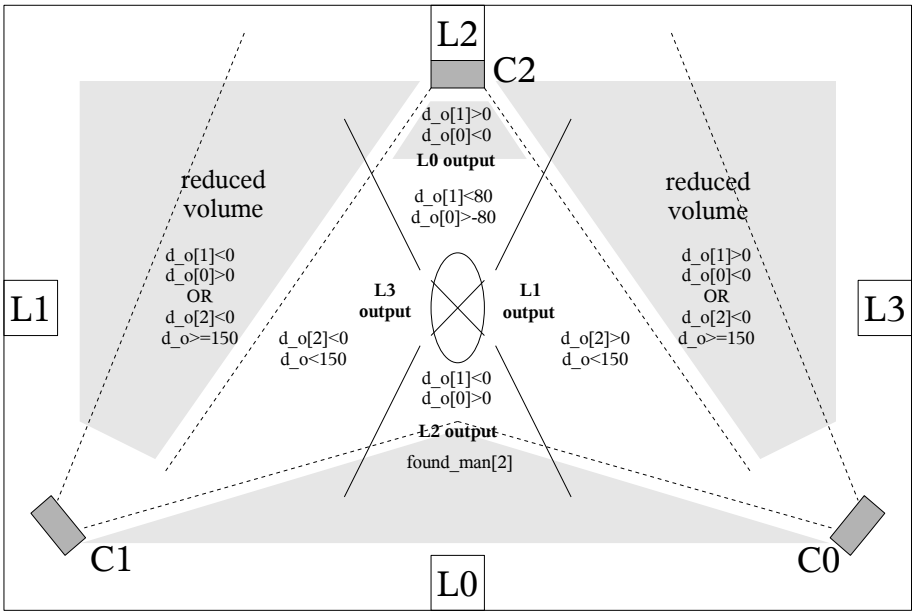

Figure 2: Diagram summarizing the conditions for the generation of each possible output. (Note: `d_o` and `d_o[n]` refer to the respective `distance_obs` parameters in the code.)

between user and obstacle can be easily judged by measuring the number of pixels from the respective means. The threshold between the two volume levels is set at 80 pixels.

If the user is to the left or right of the obstacle but not picked up by the centre camera, he is either diagonally facing the camera (and occluded by the obstacle in the centre camera) or far away (out of the centre cameras field of view).

Determining the users north/south position is more difficult. Because the centre camera is facing downwards, if the user is behind the obstacle, he will be visible above it. If the user is in front of the obstacle, he will occlude it. The side cameras are then used to determine whether an obstacle is present in the scene and estimate the distance.

If the user is not at all visible in the centre camera, he is determined to be relatively far away from the obstacle and the output volume is reduced.

4 Evaluation

The system was tested with an obstacle in the centre of the arena by moving to each of the eight directions of the compass. A ring was drawn approximately 50 cm around the object to determine the distance within which the output sound should be the higher of the two loudness levels. The results, compared to the ground truth, are given in the table below. Please note that, in the direction of the corners of the room, either of the neighbouring speaker outputs was accepted to score a 1. If there was some noise in the output (e.g. an incorrect as well as a correct output were returned), the score given was 0.5.

		valid		actual		score	
	Direction	Spkr no, `t`	Vol, `v`	Spkr no, `t`	Vol, `v`	Spkr no, `t`	Vol, `v`
inner	N	0	-200	0	-200	1	1
	NE	0,1	-200	1	-200	1	1
	E	1	-200	1	-200,-1500	1	0.5
	SE	1,2	-200	2	-200	1	1
	S	2	-200	2	-200	1	1
	SW	2,3	-200	2,3	-200,-1500	1	0.5
	W	3	-200	3	-200	1	1
	NW	3,0	-200	3	-200	1	1
outer	N	0	-1500	1	-200	0	0
	NE	0,1	-1500	1	-1500	1	1
	E	1	-1500	1	-1500	1	1
	SE	1,2	-1500	2	-1500	1	1
	S	2	-1500	2,-1	-1500,-10000	0.5	0.5
	SW	2,3	-1500	3	-200	1	0
	W	3	-1500	-1	-10000	0	0
	NW	3,-	-1500	3	-1500	1	1
Avgs						0.84	0.72

As the results show, the direction was detected more accurately from the camera information than distance. Additionally, when the person was nearer to the obstacle, the direction was more accurately detected. If the direction was returned with a `t` value of -1, this means that the person was not detected in the scene, either due to limited robustness of the tracking algorithm (in the case of south/outer) or because the person was outside of the cameras' field of vision (in the case of west/outer).

The system was found to work if the obstacle was moved. However, due to the limited space in the arena and the cameras' limited field of vision, there is not much flexibility in the set-up of the scene. The current system is designed for this fixed configuration of cameras; if cameras are added or removed, it would require re-programming. The system could, however, be adapted to a larger room or area, allowing for increased flexibility in arranging the scene.

5 Discussion and future work

5.1 The demonstration task

This work was demonstrated on 15 April 2008. It was generally agreed by the participants that certain improvements would need to be made before the system would really be robust enough to guide a blind person through a room, and therefore be demonstrable at e.g. Beginners Day.

During the concluding discussion, several ideas were proposed referring to the task itself, which would make the demonstration more eye-catching and exciting for participants. One particularly good idea was to employ a blindfolded user together with a virtual object. The user would then try to catch (or run away from) the virtual object with help from the audio signals that are fed to him. The audience could then watch the paths of user and object on the screen, and see whether they collide. A benefit of this approach is that, as there is no real obstacle, there is a potentially smaller space requirement.

A further example is the "coffee kitchen": the user could interactively select an item on the screen with the mouse, and the system should identify the blind user, and guide him to the selected object in the scene.

5.2 Technical improvements

A number of issues were also discussed, referring to the technical implementation of the system, with possible improvements being suggested.

Firstly, it was clear that the current **tracking methods** are very crude. This was indeed not the objective of this exercise. The `CVCamShift` function included in OpenCV, for example, could be used to improve the accuracy of the tracking and to follow the object once it has been detected using the current method of chroma values. This could lead to the object's shape and size and position being more accurately detected, which could result in a better 3D model.

Another major point was that the current method of **sound output** did not always reliably relay the information back to the user. Three points were discussed here:

- First, the two loudness levels were not always easily distinguishable by the user. The use of different repeating frequencies, such as in parking sensors on modern cars, could improve the situation.

- Secondly, although four speakers were used, it was not particularly easy for the listener to determine from which direction the sound was coming. One suggestion was the use of headphones, where the 3D sound could be more accurately positioned in the user's perceptive field. This would, however, require detection of the direction in which the user is facing. It might be worth making contact with Prof. Gernot Kubin at TU Graz, who specializes in audio communication, for advice.

- Thirdly, a possible improvement would be not to generate output based on the current *state* (i.e. the current position of the user and other

objects in the scene and their relations), but on how this state changes from one frame to the next. Thus, if the user steps towards an obstacle, only then a warning sound could be generated, which would warn the user that such a move is false. Thus, the user would slowly generate a mental image of the scene whilst navigating it, much like as if he could actually see.

Finally, a few notes comments were made about the current set-up of the **cameras**. The current method of downloading the image from the Axis cameras over HTTP is not particularly efficient. Ideally, the system would run on the newest gigabit cameras available in the lab, which would be permanently mounted on the wall, so that the demonstration could be quickly set up whenever the need arises.

Appendix

The following code snippet describes the algorithm used to generate the output based upon the user's and obstacle's positions.

```
// default values; t is loudspeaker number; v is loudness
t = -1; // void (no sound)
v = -10000; // silence
// loudness levels: -1500 (quiet); -200 (loud)

// found_man[c] and found_obs[c] are true if man and obstacle
// respectively are found in camera c

// distance_obs[c] is distance between mean values of object and obstacle
// (can be negative depending on direction)

// occ_obs[2] is true if obstacle is behind man from centre (north) camera

if (occ_obs[2]==-1 || (found_man[2] && !found_obs[2]) // man in front of obs
  || (distance_obs[2] <= 100 && distance_obs[2] >= 100)) //occlusion not always detected

{
  if (distance_obs[1]>0 && distance_obs[0]<0)
  {
    t=0;
    v=-1500;
    // if close, get louder
    if (distance_obs[1]<80 || distance_obs[0]>-80)
      v = -200;
  }
}
else if (occ_obs[2]==1 || (!found_man[2] && found_obs[2])) // obs in front of man
{
  if (distance_obs[1]<0 && distance_obs[0]>0)
    //found_man[2] not a strict requirement here, though it happens in practice
  {
    t=2;
```

```
    v=-200;
  }
  else if (found_man[2])
  {
    t=2;
    v=-1500;
  }
}

if (t==-1) // not set yet; look left and right
{
  if (distance_obs[2] > 0)
  {
    t = 1;
    if (dist_obs < 150)
    {
      v = -200;
    }
    else
    {
      v = -1500;
    }

  }
  else if (distance_obs[2] < 0)
  {
    t = 3;
    if (dist_obs < 150)
    {
      v = -200;
    }
    else
    {
      v = -1500;
    }
  }
  else if (distance_obs[1]!=0 || distance_obs[0]!=0)// no result from middle camera
  {
    if (distance_obs[1]<=0 && distance_obs[0]>=0)
    {
      t=1;
      v=-1500;
    }
    else if (distance_obs[1]>=0 && distance_obs[0]<=0)
    {
      t=3;
      v=-1500;
    }
  }
}
```