Image-based Automatic Object Localisation in iSpace Environment

Péter Zanaty[1], Péter Korondi[2,3], Gábor Sziebig[4], László A. Jeni[5]

[1]Narvik University College, Department of Scientific Computing, Electrical Engineering and Space technology, peter.zanaty@gmail.com

[2]Budapest University of Technology and Economics, Department of Mechatronics, Optics and Applied Informatics

[3]Computer and Automation Research Institute, korondi@sztaki.hu

[4]Narvik University College, Department of Industrial Engineering, gabszi@hin.no

[5]The University of Tokyo, Institute of Industrial Science, laszlo@hlab.iis.u-tokyo.ac.jp

Abstract: Localization is a fundamental task in mobile robotics and in indoor environments we can use various sensors to solve this problem. In the Intelligent Space environment we can use laser range finders or ultrasonic positioning systems to localize and track mobile robots. Nevertheless, our final goal is to substitute these sensors and accomplish this task using just cameras. In this paper, we show the feasibility of determining the robot's location based on the images received from a single camera. In our experimental room we used a surveillance camera, which can be controlled to pan/tilt in order to change the point of view. The camera had been mounted on the ceiling and six preset positions were selected to cover the whole area. The object recognition is based on colour space filtering and contour detection. Finally, the contours of the detected objects are transformed from the image space to the world coordinate system and the polygons are reduced to simpler ones. The system is able to detect not only the position of the objects, but their orientations.

Keywords: iSpace, image processing, localisation, calibration

1 Introduction

1.1 Intelligent Space

The Intelligent Space (iSpace) is a concept of combining electrical devices together as distributed intelligent networked devices (DIND) for the purpose of supporting people in it [1,2]. The DINDs are processing the obtained information

in order to understand the current state of the space they are in. This information forms the basis of governing the different actuators in it (see Figure 1).

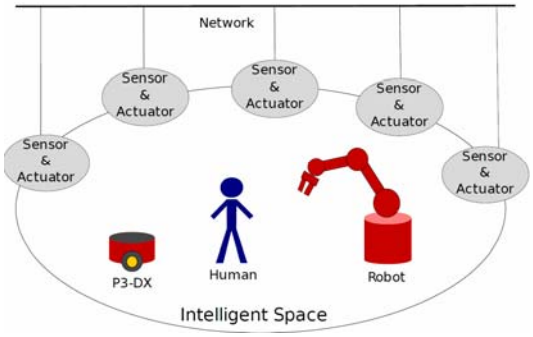

Figure 1
Intelligent Space

Such intelligent environments are able to watch what is happening in them, build a model of them, communicate with their inhabitants and act based on the decisions they make. Especially the capability of the environment to act as a context-sensitive user interface (e.g. to respond to gestures), and the reaction in certain situations (e.g. accidents, intruders) promises a range of possible application scenarios such as intelligent hospital rooms, factory, asylum for the aged, etc.

There is a number of similar research approaches to iSpace, a few of them are the following:

- *EasyLiving* project [3] at Microsoft Research is concerned with the development of architecture and technologies for intelligent environments which allow the dynamic aggregation of diverse I/O devices into a single coherent user experience.
- *Project Oxygen* at MIT aims to create a pervasive, human-centered computational environment [4].
- *The Interactive Workspaces Project* at Stanford University has a basic motive is to explore new possibilities for people working together in technology-rich spaces [5].
- *Living with Robots and Interactive Companions* project is a collaboration of 10 European partners specialized in psychology, ethology, human-computer interaction, human-robot interaction, robotics and graphical characters [6].

These projects are more or less the same initiatives targeting a different audience therefore having different portfolio. EasyLiving focuses on extending the current desktops to the homes by expanding the user experience starting from the PC.

Project Oxygen applies handheld and embedded devices to provide user centric services. The interactive workspaces project focuses on augmenting a dedicated meeting space with large displays, wireless or multimodal devices, and seamless mobile appliance integration. While the LIREC project aims to establish a multi-faceted theory of artificial long-term companions relying on results of social sciences such as etymology. On the other hand the concept of iSpace is more robotics and industrial technology inspired.

There is a considerable effort of research on sensor networks, mobile robot control, and Ubiquitous Computing [7], which also is more or less related to the iSpace initiative.

Often mobile robots are introduced into an intelligent space as actuators. Tracking humans and tracking and controlling robots in iSpace are complex tasks, for a review of current research on this topic related to iSpace see [8].

In this paper yet another possible extension onto the concept is discussed: remote control and interaction with an iSpace. Sensory data may need to be transmitted to a remote site for processing and control data may need to be sent back to agents in an iSpace. Moreover, remote control by humans of a distant iSpace may be necessary and optimal. Possible application scenarios of telemanipulation include:

- operating in domains which are not possible by simple means (e.g. micro and nanomanipulation)
- handling tasks in hazardous environments (e.g. subsurface, space or nuclear plant)
- specialized work on distant locations (e.g. brain surgery)

1.1 Narvik iSpace Room

At Narvik University College a temporary iSpace concept room was established, for the purpose of iSpace related experiments (see Figure 2).

Figure 2
Narvik iSpace room

The room originally is designed for testing heating equipments and measuring the gas by products in various ventilation conditions. To be able to handle the interaction with the reality, the localization of objects is necessary. Therefore, it was essential to install lightweight, easy to setup - easy to remove, extension to the room to meet the needs of creating a temporary test room for intelligent space.

2 Object Recognition via Surveillance Camera

This section deals with the details of object recognition at Narvik iSpace concept room. The best solution for equipping the room with object recognition facility was to use web camera. The room is approximately 4.8 meters long and 3.6 meters deep with the height of 2.5 meters, this means that at least 6 normal (60 degree field of view) cameras are needed to cover the whole room. As six cameras were not available, the only solution was to use a surveillance camera, which can be commanded to pan/tilt in order to change the point of view.

The actual camera model is the WVC200 from Linksys which is designed for home or small companies to survey their properties while they are not present.

Using the libCurl library a camera controller has been created, which can control the camera via http requests. The controller also requests the live MJPEG stream from the camera.

The MJPEG header is found out to contain the following useful information: total size of the frame, width and height of the frame, offset of the chunk, size of the chunk data and a timestamp. According to that a stream capturer is constructed, which, using the FreeImage library, parses the stream from the camera and transforms it to raw bitmap frames.

2.1 Model of the Camera

The camera had been mounted on the ceiling of the room in the central part, and six presets were selected to cover the whole area of the room. These presets can be reprogrammed automatically using the created camera handler. The area visible by the cameras in this setup is visible at Figure 3.

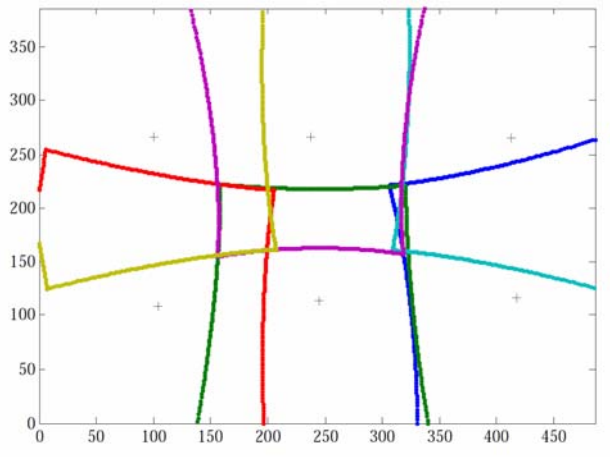

Figure 3
Six camera presets covering the whole area of the room

To use the raw data from the camera the mapping between the actual pixels and their position in the real space had to be done. There are numerous papers dealing with camera calibration, one popular method is to use Tsai's camera calibration. The camera is calibrated according to a relaxed version of the proposed method in [9]. The following parameters represent a given state of the camera:

- camera location at $Eye(Eye_x, Eye_y, Eye_z)$
- central pixel of the camera is mapped to $At(At_x, At_y, At_z)$
- the direction which is considered up vector by the camera is $Up(Up_x, Up_y, Up_z)$
- resolution of the camera is $640x480$ which is constant
- respective focal length for x and y axis fc_x and fc_y
- center of the distortion $cc(cc_x, cc_y)$
- the radial camera distortion coefficient[1] kc

The method which transforms a point from the real space to the cameras picture can be divided into two steps, first is the extrinsic transformation which transforms the global coordinates to camera coordinates. Then the intrinsic transformation transforms the camera coordinates to actual pixels.

[1] Experiments have shown that the tangential distortion is negligible, and using only the radial distortion results in a more cleaner model in terms of implementation.

The extrinsic transformation (projection) is the following:

$$(x_c, y_c) = \left(\frac{x_0}{z_0}, \frac{y_0}{z_0} \right), \text{ where } (x_0, y_0, z_0) = (x, y, z) R^T \quad (1)$$

$$\text{and } R = \begin{pmatrix} S^T \\ U^T \\ -F^T \end{pmatrix}, \text{ where } S = \frac{F \times Up}{|F \times Up|}, F = \frac{At - Eye}{|At - Eye|}, U = \frac{S \times F}{|S \times F|}, Up = \frac{Up}{|Up|} \quad (2)$$

where x_c, y_c are camera coordinates and x, y, z are the global coordinates.

As we can see this transformation is not invertible, but if we know the distance of the point to the camera, or for instance the object is located on the floor, then we can also make the inverse transformation.

The intrinsic transformation (radial lens distortion) is calculated the following way:

$$x_i = cc_x + x_d * fc_x \text{ and } y_i = cc_y + y_d * fc_y \quad (3)$$

$$\text{where } (x_d, y_d) = (x_c, y_c)(1 + k_c * r_c^2) \text{ and } r_c^2 = x_c^2 + y_c^2 \quad (4)$$

and x_i, y_i are actual pixels and x_c, y_c are from (1).

To invert this transformation we need to be able to calculate the original radius from (x_d, y_d). Luckily the square of radius of these coordinates $(r_d = \sqrt{x_d^2 + y_d^2})$ and the square of radius of the original coordinates $(r_d = \sqrt{x_c^2 + y_c^2})$ satisfies the following equation:

$$r_d = r_c * (1 + k_c * r_c^2) \Rightarrow r_d = k_c * r_c^3 + r_c \quad (5)$$

This equation should have only one solution which is less than one. Normally it will have three solutions: one which is negative, one which is larger than one, and one which will be our the solution. This is deductive from the root properties of this specific type of cubic equations. This means that solving this equation with the use of Cardano's formula results in being able to invert this transformation.

In order to calibrate the camera a grid with the distance of 40 centimetres between adjacent points has been recorded from the different presets. Knowing both the real positions of the grid points and the image positions, a Matlab script was created as to find out the ideal parameters for the model described above. The benefit of using this model rather than normal projection is visualized for one preset with Figure 4.

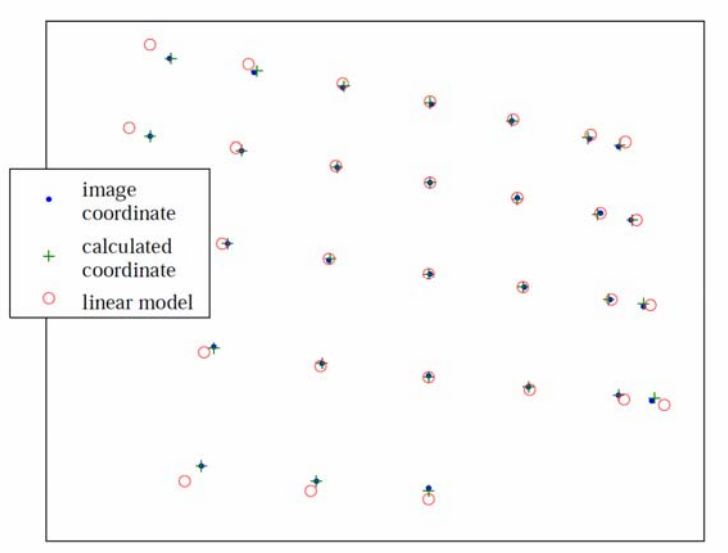

Figure 4
The lens distortion camera model

To calculate the parameters a Matlab program has been created. It tries to minimize the error of the model, and can find adequate parameter results.

2.2 Automated Position and Orientation Recognition

The camera handler is capable to instruct the camera to go to a given preset, the response time for these commands is a varying, so detection whether the camera has reached the position is vital. Considering some physical properties, it can be assumed that the camera cannot change its location in less time then T_1. Similarly, we can assume that if it was operating correctly, then the worst case response time for the camera to reach its position would not exceed T_2. This means that the camera should detect the image getting stable in $[T_1,T_2]$ time-frame. Detecting if the image is stable can be done with calculating average absolute difference between consecutive frames using the OpenCV library.

Object recognition from visual data is harder than one might notice. Human visual processing is one of the most advanced systems known in the universe [10]. Pretending that we can automate the identification of objects easily using a camera and a computer is pointless. Instead we can make a rational constraint to deal only with certain objects.

To able to differentiate these objects from each other and their surroundings, labels with different colours are used. To provide a facility with which labelled

objects can be recognized the OpenCV library is used. The introduced object recognition process is the following:

- filter the image in colour space (what is in the same colour as the label)
- detect the contours (where are possible objects)
- transform the contours into real coordinates (remove camera distortion)
- filter the contours (does the contour looks like what we expect)
- smooth the contour (if we want to detect orientation)

The OpenCV library provides routines to filter an image in HSV[2] colour-space then to detect the contours in it. In order to separate the noise which could be recognized from the real objects, these contours should be filtered. First the area of the contour is calculated then a bounding box is aligned to the contour. Using these it is possible to filter out most of the noise and only get the contours of desired objects. These contours are transformed from image coordinates to real coordinates, and then the polygons are reduced to simpler ones (typically from around 150 vertex to 3 or 4 vertex). To be able to detect not only the position of these objects but also the orientation a special isosceles triangle shape is used, which has a considerably smaller side and two larger sides.

To describe an object (label) the following data is needed (see Table 1):

Property	Type	Description
rangeNo	integer	Number of colour filter ranges
range[]	CvScalar[2]	HSV range endpoints of the colour filter
area	float[2]	the minimal and maximal area of the object in cm^2
sideRatio	float[2]	ratio of the sides of the object
width	float[2]	minimal and maximal width of a bounding rectangle
height	float[2]	minimal and maximal height of a bounding rectangle
objHeight	float	estimated height of the object from the floor
x	float[2]	min. and max. x coordinate of the bounding rectangle
y	float[2]	min. and max. y coordinate of the bounding rectangle

Table 1
Properties describing recognizable objects

[2] HSV (Hue Saturation Value) is a color-space which describes the perceptual color relationships; using HSV in color filtering results in more intuitive boundaries

These results had been combined into an automated camera system, which can change its position between the presets automatically and seek the whole room for recognizable objects. If any object it detected, the iSpace framework will be notified through the predefined interfaces along with the position and orientation data.

3 Camera System Test

The camera system is described in Section 2 and the applied model is discussed in details in Section 2.1. For an illustration of the test room see Figure 5.

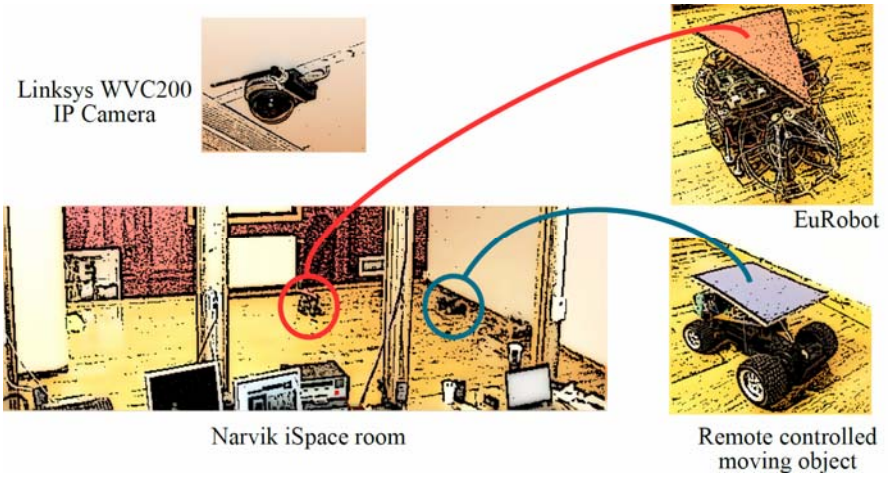

Figure 5
The test room in Narvik, Norway

To assist and verify the object recognition module, a graphical interface has been created. The graphical interface serves multiple purposes.

One of the purposes is to test the communication between the module and the camera: the picture coming from the camera can be viewed directly by using this GUI.

The second option is to command and calibrate the presets of the camera via this tool. Using these commands one can navigate the camera with the camera tester module, and see if it is working correctly. Also it is possible to record the presets automatically, if the camera does not know them yet or for some other reason the presets have became corrupted.

Third purpose of this GUI is to help camera calibration. When the camera model parameters are not yet adjusted to a room or for some reason these properties have

changed (e.g. the position or the direction of the camera has been slightly changed), calibration is essential. Calibration of the camera can be broken down to three steps.

- First we need to place distinctive markers on the floor (or anywhere in the room what is visible by the camera at the given preset) and measure the absolute positions of these markers.
- Second step is to get the pixel coordinates of these markers from the picture recorded by the camera at the given preset.
- Third step is to calculate the camera parameters.

The GUI itself provides a facility to record these pixel-coordinates from a given preset, by magnifying the picture under the mouse cursor, and recording the coordinates of the mouse clicks.

The fourth function of the GUI is to aid the object recognition process by gathering colour information from the camera. There are two functions to ease the creation of HSV ranges. The first is when the user clicks with the right button on the picture the application prints out the colour of the pixel and displays a window of the colour to the user. The second feature is to be able to specify a polygon on the picture which will be the region of interest, and then calculate the histogram for it. This histogram holds the prevalence of the given hue-saturation domains of the selected region (see Figure 6).

(a) Light green (b) Red (c) Dark blue

Figure 6
Histograms made by the graphical camera toolkit

The fifth function is to give visual feedback about what objects are recognized and where. For this task there is a dedicated window which represents the room and whenever an object is recognized it is drawn on to it. When the camera moves to a new position the previous recognized objects are shaded, this way old recognitions disappear over time.

Using this graphical interface both the camera and the underlying image processing faculties can be tested real-time. A printer friendly modified screenshot showing the recognized objects mapped to real coordinates is visible in Figure 7,

this picture illustrates, that the applied camera model's inverse transformation can accommodate the real sizes and the real shapes of the objects quite well.

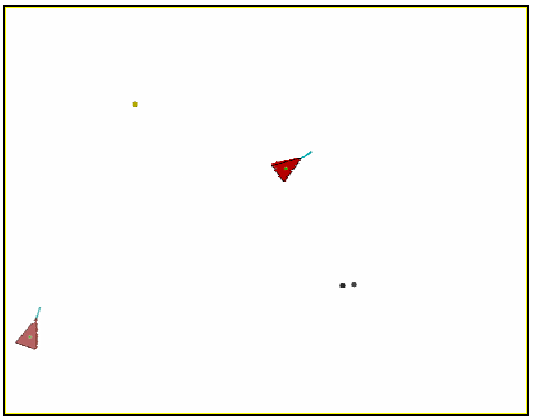

Figure 7
Image processing in action: modified screenshot showing recognized objects at the iSpace test room in Narvik, Norway

Conclusions

In this paper we have presented a solution for mobile robot localization in the Intelligent Space environment. We showed that it is feasible to achieve indoor localization using one pan/tilt surveillance camera and no additional infrastructure is required. For the object recognition we used the OpenCV library and the solution is based on colour space filtering and contour detection. The robots do not have to be tagged or carry extra devices.

Results can be improved by using multiple cameras with higher resolutions. Concerning future work, we would like to use a three-camera system, where each camera covers a smaller area and the image processing can be done in parallel.

Acknowledgement

The research was supported by ETOCOM project (TÁMOP-4.2.2-08/1/KMR-2008-0007) through the Hungarian National Development Agency in the framework of Social Renewal Operative Programme supported by EU and co-financed by the European Social Fund and the National Science Research Fund (OTKA K62836).

References

[1] Joo-Ho Lee and Hideki Hashimoto. Intelligent space – concept and contents. Advanced Robotics, 16(3):265–280, 2002

[2] Peter Korondi, Hideki Hashimoto, "Intelligent Space, as an Integrated Intelligent System", Keynote paper of International Conference on Electrical Drives and Power Electronics, Proceedings pp. 24-31. 2003

[3] Barry Brumitt, Brian Meyers, John Krumm, Amanda Kern, and Steven A. Shafer. Easyliving: Technologies for intelligent environments, in Proc. of Second International Symposium on Handheld and Ubiquitous Computing, pp. 97-119, 2000

[4] Larry Rudolph. Project oxygen: Pervasive, human-centric computing - an initial experience. In CAiSE, pages 1–12, 2001

[5] Brad Johanson, Armando Fox, and Terry Winograd. The interactive workspaces project: Experiences with ubiquitous computing rooms. IEEE Pervasive Computing, 1(2):67–74, 2002

[6] LIREC: LIving with Robots and InteractivE Companions, homepage. [Online] http://www.lirec.org/

[7] Mark Weiser. Some computer science issues in ubiquitous computing. SIGMOBILE Mob. Comput. Commun. Rev., 3(3):12, 1999

[8] Drazen Brscic and Hideki Hashimoto. Model based robot localization using onboard and distributed laser range finders, in Proc. of IROS, pp 1154–1159, 2008

[9] Berthold K.P. Horn. Tsai's camera calibration method revisited. Technical report, MIT Artificial Intelligence Laboratory, 2000

[10] A. D. Millner and M. A. Goodale. The Visual Brain in Action. Oxford University Press, 1998