Piotr Skrzypczyński, PhD
Technical University of Poznań, Institute of Control and Information Engineering

AN APPROACH TO LOW-COST REAL-TIME VISUAL PERCEPTION IN A MOBILE ROBOT

Abstract : This article outlines the development of a low-cost vision system for a small indoor mobile robot. The system is based on the of-the-shelf hardware components and open source software. It uses a webcam connected to a standard PC through USB as the sensor and the CMVision library available for free as the core image processing module. Visual perception is applied to the problem of target identification and tracking in the behaviour-based navigation framework. Results of experimental evaluation of the complete system are presented. Some remarks on other possible applications of the developed vision system are given.

1. INTRODUCTION

To exhibit a significant level of autonomy a mobile robot must have abilities to recognize appropriate features of the environment. Different sensing modalities and many particular sensor designs have been proposed in the literature [5]. Among the sensors that are used on mobile robots, one of the most powerful is the machine vision. Vision is popular in robotics due to the availability of the hardware (cameras, unlike e.g. laser scanners are used in many other, much more commercial applications) and the inherent power and generality of the visual sensing, which is also the main sensing modality in most of the animals, including the human beings. However, machine vision has proved to be difficult in practical implementation on the mobile robots due to the complex processing methods and high computing power needed to obtain perceptual information required in the navigation tasks. Also the information processing and representation paradigms widely used in the computer vision and AI community [7] turned out to be much less appropriate in the mobile robotics domain. Mobile robots are situated agents, which must take into account both the uncertainty and the dynamics in the surrounding environment. An approach to the visual perception, which has proved successful on many mobile robots is to specialize the image processing to support particular robotic tasks such as obstacle avoidance, object tracking and self-localization by separate modules. This paradigm has been introduced together with the behaviour-based architectures [1], resulting in mobile robots which, although rather simple, are able to "survive" in the real world by using visual perception. However, an obstacle in the wider implementation of the machine vision on mobile robots remains the high computing power required to process the video stream in real time. Many researchers use specialized hardware to implement their vision algorithms in real time [2]. A hardware-based approach has several disadvantages: it significantly increases the costs of the sensor, and usually causes the researcher to use software and development tools, which are proprietary to the hardware manufacturer.

These considerations are especially important for low-cost mobile robots used by universities and other educational institutions.
This article describes the results of a project aimed at developing a low-cost, general purpose vision system for the *Pioneer 2DX* mobile robot used at the Mobile Robotics Lab at TU Poznan, mainly for educational purposes. Our design considerations for the project were the following:

- to develop a system supporting behaviour-based navigation by recognizing in real time selected objects (targets, landmarks) marked in an unobtrusive way;
- to use a cheap, of-the-shelf camera, which can be easily integrated on the robot;
- to avoid use of any specialized hardware and proprietary software;
- to provide an easy access to the high-level perceptual information for the particular programs controlling the robot;

To meet these requirements, the robot has been equipped with an internet-type camera connected to the on-board computer by the USB interface. The image processing is based on segmentation in the colour space. After experiments with several segmentation algorithms, we have decided to use for the low-level image processing the CMVision [4] – an open source colour segmentation library.
The following sections describe the motivation to use a webcam on the mobile robot, the image processing methods employed in the system, the integration of the vision system with the robot control programs, and the experimental results achieved. The article concludes with some remarks on the planned future extensions to the described system.

2. WEBCAM AS A SENSOR FOR MOBILE ROBOTS

At the very beginning of the described project we have faced a problem: how to provide the *Pioneer 2DX* small-size mobile robot with a colour vision system at very low cost, avoiding also any substantial mechanical and electrical modifications to the robot ?
Although the CCD technology made the cameras small and simple enough to be used on mobile robots, the industrial-type colour cameras are still much more expensive than their consumer market counterparts manufactured in high volumes. Such cameras having analog output require also additional frame grabbers to interface them to a host computer. Although there are low-cost, universal frame capture devices available (e.g. *BrookTree BT848*-based), not all mobile robots can use them. The *Pioneer 2DX* has a PC-104 standard single board computer, which does not have any PC-compatible expansion slots. The manufacturer of the *Pioneer* offers a vision system for this robot using a PC-104-based frame grabber, but this system with a pan/tilt/zoom colour camera is rather expensive [12].

Recently, digital cameras, which communicate to the computer through a standard interface (typically USB or IEEE 1439) became available, what eliminates the need for a separate frame grabber. In this category of devices, the internet cameras (so-called "webcams") are particularly interesting for use in a low-cost vision system for a mobile robot. These cameras have been designed to provide images and video streams for multimedia applications (particularly the WWW browsers) on personal computers. A breakthrough in this group of vision sensors was the advent of very cheap CMOS-technology-based, highly-integrated cameras, available at prices below 50 Euro.

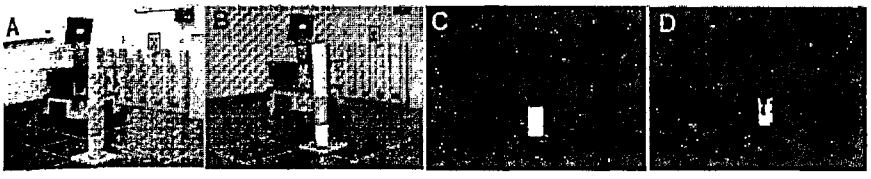

Figure 1: Example images taken with the PCVC740K (A) and the CMOS-based MT404 (B) and the results of colour-based segmentation of these images

After an analysis of the available solutions, we have decided to use a webcam on the *Pioneer 2DX*. Such a camera is very small, lightweight, connected through the USB, which is readily available in the on-board computer, and does not need any additional power supply. Internet cameras have been already used in mobile robotic applications, an example of a successful system is the W-CAPS absolute positioning system using several wall-mounted webcams to determine a mobile robot position with centimeter-level accuracy [8].

After first tests of several low-end webcams (e.g. *MediaTech MT404*) we were a bit disappointed by the poor quality of the images, especially in low light conditions. We have found, that the cheap CMOS-based cameras demonstrate all the weaknesses typical to this technology. The advantage of CMOS is that it can be fabricated at much lower cost than CCD devices. The CMOS-based cameras require also less power. However, they have higher stationary noise than found in a CCD and reduced light sensitivity in comparison to a similar size CCD matrix. A problem in the CMOS cameras is also their high sensitivity to the infra red (IR) light. In cheap cameras having no IR-blocking lenses, the images acquired in environments lit by light bulbs or a direct sunlight are very red while other colours fade to grey.

Finally, we have found the *Philips PCVC740K* camera, which fulfilled our requirements as to the image quality (for about 80 Euro). This camera uses a 4.8 mm F3.0 lens, a 1/4" CCD sensor, and can provide images in VGA (640 × 480) resolution and 24-bit colour. The images obtained from this camera have much lower background noise and more intensive and stable colours than images from the low-end CMOS-based models. The camera works well even in quite low light conditions (¿ 2 lux). What was very important to the project, the PCVC740K has very good

support for the Linux operating system, which is used on the *Pioneer 2DX* robot. It works with the Video4Linux drivers [15], which are integrated into the kernel (version 2.2.x or higher), and provide a device-independent high-level interface to the video capturing devices. Although Video4Linux does not support some features specific to the internet camera (it is rather oriented toward video/TV cards), this functionality can be controlled through the native Philips webcam driver for Linux [14], which allows to modify some default settings for the camera (image format, frame-rate, compression), and to control the gain. The camera has an auto gain function, which adjusts the gain on the R, G and B channels equally. However, the changing brightness is undesired when the camera is used to detect objects of a specific colour, so the auto gain is turned off.

The Philips webcam driver provides colour images in the YUV 4:2:0 planar (YUV420 format. The YUV colour standard consists of an intensity value and two chrominance values. This format is commonly used in video capturing devices because it complies to the TV colour palette (e.g. PAL). In the YUV420P format there is one U and one V value for every four Y values, so each U and V value apply to a block of 2×2 pixels.

We are able to get up to 30 frames per second in the 320×240 resolution from the PCVC740K connected to the robot's Pentium III 800MHz computer through the USB link. This is enough for most of the applications of this simple vision system. So, we can say that the internet camera (of a good quality) is a satisfying solution to the problem described at the beginning of this section.

3. COLOUR-BASED IMAGE PROCESSING

Using the low-cost webcam, we had to take into account its limitations in the image processing software. Although the images with good contrast and live colours are visually appealing, they have lower resolution and higher noise than images obtained from a professional colour camera connected to a frame grabber. Also the depth of focus is quite small. Due to these properties of the hardware, we did not try to implement edge detection. We rather focused on the colour-based image processing. We choose the colour-based region segmentation to identify particular objects in the robot environment. This approach has already proved successful in many mobile robots, particularly, it is main visual sensing modality in the robotics soccer. Also the ActivMedia Robotics offers a colour tracking software for their *Pioneer* robots. However, this is a commercial software working only with the BT848-based frame grabbers, and not available in source code form [13].

The aim of the image processing was to find in the image the objects having specific colour. This task had to be accomplished in real-time, i.e. at the frame rate of the vision system, using only the computational power of the on-board PC of the robot. This makes clear, that the simplicity of the algorithm and optimization of the code were the most important issues.

The first step was to choose the right colour space. The RGB (Red-Green-Blue) representation commonly used to display images has the intensity coded into all three components, what makes the colour tresholding difficult. The HSI space

(Hue-Saturation-Intensity) has much better properties w.r.t. the colour-based processing, because it has a separated intensity component and the chrominance is coded in the remaining two (H,S). However, the conversion to HSI is difficult [6], and when the saturation value is low, the hue component is not reliable for discrimination (tresholding), what can lead to some instabilities in the segmentation. The PCVC740K camera natively provides output in a variant of the YUV colour space, which also has the separated intensity component (Y) and two values (U,V) describing the chrominance. To avoid additional conversion and potential problems with the HSI model, we have decided to use the YUV space for the colour tresholding.

To find in an image pixels belonging to the objects specified by their colour, we use simple tresholding in the YUV space. A colour is defined as a set of constant thresholds in a form of a cubic structure in the colour space [7]. This method is simple enough to run in real-time on a PC, providing that the code is carefully optimized. After first tests of the feasibility of this approach (results are shown in Fig. 1) for the segmentation of the webcam images, we have decided to use the CMVision library [4] as the basic layer of our image processing software. CMVision provides highly optimized implementation of the threshold-based classification algorithm. It is a free software for Linux, and can be extended and modified to meet particular requirements due to source code availability. This library uses the Video4Linux standard and YUV as the native colour space, what makes it a perfect match to our low-cost vision system. By using the CMVision it is possible to obtain information on the centroids and approximate sizes (in pixels) of the found regions.

To use the library with the PCVC740K webcam, we have implemented a procedure, which converts the frames coded in the camera's native YUV420P sub-format into the YUV422P colour space required by CMVision. The YUV422P format differs from the YUV420P only in having two, instead of four, Y values per one U and V value, hence the conversion procedure does not add significant overhead in the frame processing.

4. INTEGRATION IN THE NAVIGATION SYSTEM

The first application of the webcam-based vision system on the *Pioneer 2DX* robot is goal seeking in the behaviour-based navigation architecture. Behaviour-based architectures involve decomposition of the whole navigation system into a number of simple behaviours. Each behaviour produces actions aimed at achieving a particular objective, such as obstacle avoidance, goal seeking or path tracking. For practical implementation of individual behaviours fuzzy logic is often used. Fuzzy behaviours are simple, do not require analytical model of the robot, and are robust to the uncertainty, which is inherent in mobile robot control [9]. In [11] we have presented a fuzzy-logic-based controller for the mobile robot. The robot uses if-then rules to navigate from a given start position to a given goal position, while avoiding collisions with obstacles. The goal point has been described by its x_g, y_g coordinates in the global frame. Because manual design of many reactive

behaviours can be quite complicated and tedious task, we have approached the problem of fuzzy behaviour learning by using the classifier system with genetic algorithm. The fuzzy rules are evolved in a realistic simulation, then transferred to real mobile robots for further testing.

The *Pioneer* robot is used to perform most of the experiments with the fuzzy behaviour-based control. These experiments have shown critical role of reliable robot localization. The navigation is purely reactive, the robot does not perform path planning, hence it should not depend heavily on the accuracy of its localization system. However, the input vector of the fuzzy controller $\mathbf{X} = (\beta_r, l_1, l_2, \ldots l_n)$ consists of the distances to obstacles l_i given by n range sensor groups and the orientation β_r of the robot relative to the goal. When the goal is defined only by its global coordinates, the β_r value is computed by relating the position and orientation of the robot (the pose) given by the odometry to the position of the goal in the global frame. Whenever the pose provided by the odometry diverges significantly from the real one, the robot can turn in a wrong direction being not able to achieve the x_g, y_g point.

To solve this problem, we have decided to use the developed on-board webcam to track the goal. The goal is no longer just a point in the x, y coordinates, it is a physical object (called a target), which can be distinguished and tracked by the vision system. As we use colour-based image segmentation, the target can be any object having a distinctive colour, e.g. a tube with a patch of colour paper or a small, bright-coloured toy. When visual tracking is used, the orientation of the robot relative to the goal is the egocentric angle to the target object, computed from the position of the centroid of the tracked object in the camera image:

$$\beta_r = x_t \frac{\alpha}{R_x}, \tag{1}$$

where x_t is the shift of the target centroid from the center of the image, R_x is the horizontal resolution of the camera, and α is its viewing angle, which has been determined experimentally. Hence, whenever the tracked object is in the field of view of the on-board camera the β_r is computed independently of the current robot pose given by the odometry. This allows the robot to turn toward the goal even if it does not know exactly its pose.

A problem arises when the target is temporarily not visible, e.g. when it is occluded by the obstacles. In such a case the robot can use the memorized global coordinates of the goal to compute the β_r. Because the current robot pose in the global frame is a subject to errors introduced by the odometry, the goal coordinates x_g, y_g are re-computed every time the robot sees the target. Because the x_g, y_g values are used to estimate the current β_r angle and only the relative position of the robot w.r.t. the goal is needed, the position of the target in the global frame is meaningless. Whenever the robot can not see the target it uses its last estimate of the goal position. Hence, the computed β_r is corrupted only by the odometry errors accumulated since the last observation of the target. This error is much smaller than the robot pose error w.r.t. the global frame. To compute the position of the target w.r.t. its current pose the robot uses the egocentric angle to the

target, and the distance between the camera and the target. The angle is available from the vision system, but to compute the distance the robot needs to have some additional knowledge about the target object. From the segmentation procedure the robot knows the rough shape of the target object. If the real dimensions of the object are known it is possible to compute the distance d between the robot and the target:

$$d = \lambda(\frac{h}{h'} + 1), \tag{2}$$

where h is the known height of the actual object, h' is the height of the target on the image, and λ is the focal length of the camera known from the calibration procedure. Because the width of the observed shape of the target can change significantly depending on the aspect at which the object is observed, we use only the height of the target to compute the distance. The structure of the software system enabling the mobile robot to get to the visually distinctive target while avoiding collisions with obstacles is shown in Fig. 2.

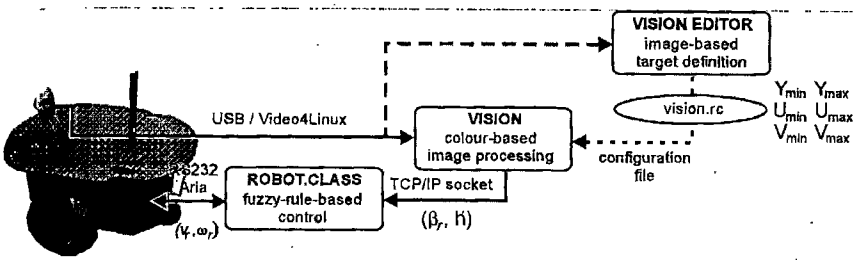

Figure 2: Structure of the robot software

One of the important design considerations of the vision system was to provide an easy access to the high-level information for the different programs controlling the robot. Because of that, the image processing module has been implemented as a separate program communicating with the control programs of the robot by means of a TCP socket. The colour-tracking functionality has been encapsulated in a server which provides to the client programs the egocentric angles to the detected objects and the rudimentary characteristics of their shapes. The client controlling the robot can be implemented independently from the vision-related part of the system, e.g. using a different programming language. Because both programs run on the same computer, the overhead due to the TCP communication is negligible. The control program for the robot has been written in Java. It uses the SWIG-based Java wrappers to the ActivMedia's *Aria 1.3* library to control the robot. The fuzzy-rule bases learned in simulation are used. The distances to the obstacles are given by eight sonars, while the information on the target position is provided by the vision server. The control program makes use of the threading capabilities of Java, by establishing concurrent threads to read the sonar sensors and to receive the information from the vision system. The vision thread computes also new

x_g, y_g values every time it receives the target parameters. A Java-based program is portable, and due to its object-oriented structure can be easily modified and extended [10].
There is a separate program (called editor), which enables the users of the vision system on the *Pioneer* robot to interactively define the colour which is searched by the segmentation algorithm. This program makes use of the XWindows GUI. The colour is defined by setting the minimum and maximum values for the Y, U and V components, which define the necessary block in the colour space. This program enables also the user to set the actual height of the target object, which is used to compute the distance from the camera to the target.

5. EXPERIMENTAL RESULTS

To test the developed vision system several experiments have been carried out. We have tested the capabilities of the camera and the segmentation software against various target objects, ranging from simple patches of colour paper to plastic toys having quite complicated shapes. Figure 3A shows the robot with the PCVC740K camera and one of these toys – a yellow chicken. This target, being rather small and having a strong texture (this is a soft and "hairy" toy) has been reliably tracked over a range of distances under natural and fluorescent lighting (Fig. 3B,C). The tresholding in the YUV space has correctly discriminated pixels having the selected colour from the background (Fig. 3D).

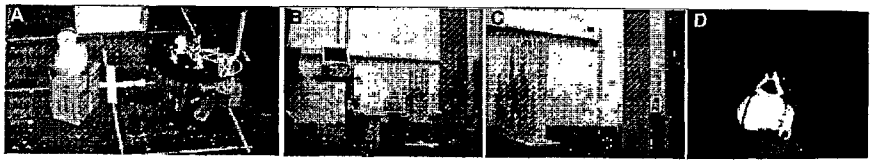

Figure 3: Results of an experiment – a toy chicken was used as the target

Figure 4 provides a sequence of images from the overhead camera mounted to the ceiling of the laboratory room, showing an experiment with the *Pioneer 2DX* robot using the vision system. The robot travelled to the target (marked with the arrow on the image) being a tube with a patch of bright-red paper attached. The overhead camera has been used only to obtain images of the scene for the documentation purposes.
The robot tracked the target reliably, but when it was occluded by other obstacles (other tubes) the robot used the estimated target position to compute the angle to the goal. Figure 4 shows also a screenshot of the editor program with the original image and the found red patch on the tube.

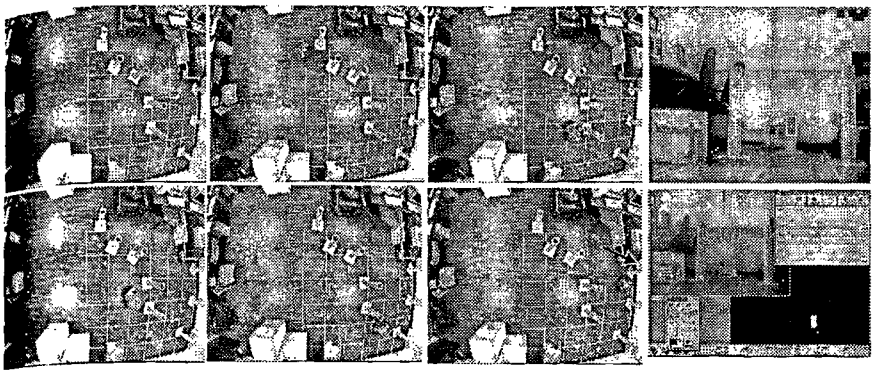

Figure 4: Results of an experiment – a red marker was used as the target

6. FUTURE WORK

Target identification and tracking in the reactive navigation system does not exhaust the possible application of the developed vision system. Because the vision server can provide angles and distances to known, colour objects, it is possible to use it for self-localization, as it is practiced in the robotics soccer domain.
We have already examined this approach experimentally, using two colour patches on the tube-like obstacles as landmarks (one of such landmarks is shown in Fig. 1). In this case the distance between the camera and the landmark is assumed to be proportional to the distance between the centroids of both colour patches on the tube. The preliminary results have shown rather poor quality of the obtained position estimates, due to the low accuracy of the angle and distance computed directly from the segmentation results. When these values are used in the fuzzy-logic-based navigation system the limited accuracy is much less deteriorating, due to the inherent robustness of the fuzzy controller to such uncertainties.
Currently, we are working on the implementation of the localization method presented in [3] on the *Pioneer 2DX* robot. The CMVision is employed to quickly find the regions of interests (ROIs), which can contain the artificial landmarks.

7. CONCLUSIONS

We have presented an approach to build a low-cost vision system for an indoor mobile robot. The system uses of-the-shelf computer hardware, originally intended for the multimedia and Internet applications, and free, open source software. In spite of that, it is able to work in real-time, providing the robot control program with reliable information on the location of selected colour objects. An example implementation of a behaviour-based navigation architecture utilizing this visual information to track the target objects has been presented.
We believe, that the presented approach can be easily followed by others (cost of

the whole system is below 100 Euro), being beneficial to both the research and education.

References

[1] Arkin R., *Behavior-based Robotics*, The MIT Press, 1998.

[2] Barnhart C., *The MIT Cheap Vision Machine*, http://www.ai.mit.edu/people/ceb/cvm.html

[3] Bączyk R., Kasiński A., Skrzypczyński P., *Vision-Based Mobile Robot Localization with Simple Artificial Landmarks*, Prepr. 7th IFAC Symp. on Robot Control, Wroclaw, 2003, 217-222.

[4] Bruce J., Balch T., Veloso M., *Fast and Inexpensive Color Image Segmentation for Interactive Robots*, Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems, 2000, 2061-2066.

[5] Everett H., *Sensors for Mobile Robots: Theory and Application*, A. K. Peters, Wellesley, 1995.

[6] Ford A., Roberts A., *Colour Space Conversions*, http://www.wmin.ac.uk/ITRG/docs/coloureq/

[7] Jain R., Kasturi R., Schunck B., *Machine Vision*. McGraw-Hill, New York, 1995.

[8] Lilienthal A., Duckett T., *An Absolute Positioning System for 100 Euros*, Proc. IEEE Int. Workshop on Robotic Sensing, Örebro, 2003.

[9] Saffiotti A., *Fuzzy Logic in Autonomous Navigation*, Fuzzy Logic Techniques for Autonomous Vehicle Navigation, (Saffiotti and Driankov, Eds.), Physica-Verlag, 2001, 3-24.

[10] Skrzypczyński P., *Guiding a Mobile Robot with an Internet Application*, Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems, Maui, 2001, 1578-1583.

[11] Skrzypczyński P., *Experiments and Results in Mobile Robot Navigation with a Fuzzy-Genetic Controller*, Proc. Conf. AUTOMATION'03, Warsaw, 2003, 233-240.

[12] ActivMedia Robotics, http://www.activrobots.com/

[13] ACTS Vision Library, http://robots.activmedia.com/acts/

[14] Linux support for Philips USB webcams, http://www.smcc.demon.nl/webcam/

[15] Video For Linux, http://millennium.diads.com/bdirks/v4l2.htm