PDA Based Real-Time Vision for a Small Autonomous Mobile Robot

Scott Jantz and Keith L Doty

Mekatronix Inc.
URL: www.mekatronix.com
Tel. 352-376-7373
&
Machine Intelligence Laboratory
Department of Electrical and Computer Engineering
University of Florida, USA
Email: scott@mekatronix.com, doty@mekatronix.com

2002 Florida Conference on Recent Advances in Robotics
May 23-24, 2002, Florida International University

Abstract

In this paper we describe a vision system based on combining a mobile autonomous robot and a hand-held *Personal Data Assistant* (PDA). All computation for control of the robot and vision are done on the robot and PDA. The TJ Pro™ can be outfitted with a *HandSpring*™ PDA and a digital camera (*Eyemodule2*™) to create a powerful vision equipped robot. Previously, outfitting a robot with vision required a large platform or off-robot computing power. With the vision and processing power of PDAs, low cost Mekatronix robots can perform complex vision functions at a fraction of the cost of current robots. Robots with vision will no longer be confined to large university research labs, NASA or the military. Robots with vision and artificial intelligence are now available to everyone.

Introduction

The TJ Pro™ / PDA platform (**Figure 1**) provides an excellent test bed for exploring visual behaviors in autonomous robots. The PDA, programmed in C or C^{++}, provides about 16Mbyte of RAM and a graphical user interface (GUI) capability on board the robot. In addition, off-line programming on a PC with PDA download capability (conduit), permits the user the convenience of a large computer for program development and debugging.

Current robots support on-board vision systems mounted on large expensive robots ($10K-30K) with expensive camera package add-ons. Large robot platforms also create safety concerns. If a large robot runs into a wall or other object during an experiment, it can damage itself or the object. Small robots do not present such risks.

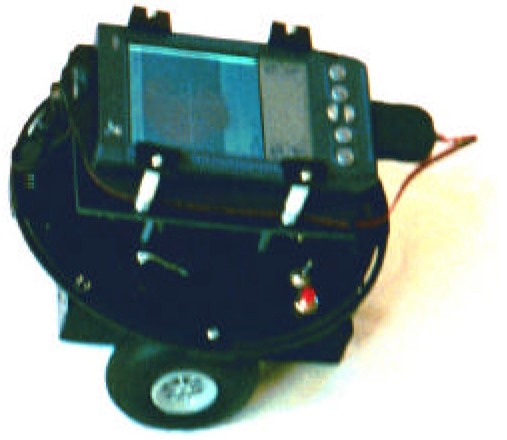

Figure 1: TJ Pro™ robot PDA vision system.

In contrast, small robots use less expensive motors, mechanical structures and batteries, greatly reducing platform costs.

The PDA graphical user interface (GUI) provides a debugging and development advantage. The screen and touch pad allows interaction with the robot while the robot is running. The interface is programmed to let us "see" what the robot sees in real-time. We can easily observe and evaluate how algorithmic output differs in various environments as well as quickly evaluating different algorithms.

Since a PDA executes slower than a desktop or laptop computer, we optimized execution time of our algorithms for this platform.

To illustrate real-time vision capability on a small, low cost robot we use two main vision algorithms, color space remapping and centroid / blob detection. Color vision allows us to simplify the environment that the robot sees. In our experiments the robot tracks a brightly colored object in a complex environment. This problem relates to soccer ball tracking in the autonomous robot-soccer games that have become popular worldwide. Our first goals were to track and intercept a soccer-like ball. In order to track a single object and not the background noise, we use a combination of centroid calculations and primitive blob detection to differentiate a red ball from, say, red woodwork along a wall. Robot ball "dribbling" toward a goal is the natural next step after ball identification and location.

TJ Pro™ Robot Platform

The TJ Pro™ platform consists of several parts. The basic TJ Pro™ robot contains a Motorola MC68HC11 processor with 32K of SRAM. The robot has 2 servomotors for locomotion, 2 IR distance sensors and 4 bump switches for contact detection. The robots power comes from 6 AA NiCd batteries. Physically the robots are 3.5 inches high and 6 inches in diameter. A platform holds the PDA 1.5 inches above the top plate of the robot and level with the floor. The interface to the Handspring PDA is through a 5V RS-232 serial port. This interface is another advantage of the Handspring PDA. The 5V serial port of the Handspring is easy to interface to the MC68HC11. The universal serial bus (USB) port on the PDA makes programming the PDA easy and fast.

Software Implementation

Software written on the MC68HC11 is written in C with the ICC11 C compiler and downloaded using the Mekatronix high-speed serial downloader (HSSDL11). The vision software written for the Handspring PDA is compiled using the GNU compiler for the Handspring and downloaded using the palm desktop tool, which comes with the PDA. The *Eyemodule* System Development Kit (SDK) provides a framework for the robot vision software and greatly accelerated software development time.

Color Space Remapping

For our robot to see a colored ball, it first must differentiate between the color of the ball and all other colors. The limited processing power of a PDA dictates efficient algorithm design, a requirement assisted by color space remapping.

RGB values produced by the *Eyemodule* camera cannot be easily interpreted as to the redness, greenness or blueness humans see. To enable the robot to differentiate between object colors roughly according to human vision, the RGB values must be remapped to another color space. Our two best options are YUV (also know as YIQ) and HSV (also known as HIS) color spaces [1] [2] [3] [6] [11] [9].
In HSV the values of Red Green and Blue are transformed into Hue, Saturation and Value where, theoretically, Hue completely represents the color independent of lighting. HSV calculations involve multiple divisions and multiplications, too computationally intensive for the PDA limited processing power.

YUV (closely related to YIQ), a color space derived from color television, preserves the chrominance of a signal while the brightness changes and simultaneously maintains compatibility with black and white color televisions [6]. Lighting changes only affect Y values. U and V values represent the Red and Blue chrominance, respectively. For the robot to track a red ball, it simply compares U and V [11]. The color mapping from RGB to YUV is

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.146 & -0.288 & 0.434 \\ 0.617 & -0.517 & -.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

This mapping normally involves floating point multiplies, consuming a substantial portion of a PDA's computation power. But, for the simple problem at hand, the robot "red-blob" detection algorithm can avoid these computations. When the RGB data is formatted 5:6:5 in a 16 bit number, the mapping becomes much simpler. The R and B values vary from 0 to 32 and G from 0 to 64. We divide green by two by discarding the LSB, thereby normalizing the dynamic range of the colors to 5-bits each. Using 32 as our max color value the color mapping equations become, after multiplying the coefficients in the matrix by 32 and rounding up to the next integer,

$$U = -5R-9G+14B$$
$$V=20R-17G-3B$$

And when U (blue) < V(red) we get

$$R>1/2G + B$$

These equations yield an extremely simple way to detect a red object.

Morphology

Morphology allows us to simplify the image and eliminate much of the noise. Morphological operators treat binary images as sets which are members of a two dimensional space [9]. For our purposes we will use the erosion operator. Erosion is defined as

$$\{x \mid (B)_x \subseteq A\}$$

Where erosion results in a set of points for which B fits entirely inside A. Erosion eliminates any pixels not part of a continuous object and most of the "shot" noise produced by the camera.

Centroid

Once the algorithm detects the ball and filters the data by the morphological transformation, the robot will know the placement of the ball in the image frame. The X and Y centroids are calculated from

$$Y = MAX_{j=0}^{j=Y-1}\left(\sum_{i=0}^{i=X}(f(x,y))\right)$$

$$X = MAX_{i=0}^{i=X-1}\left(\sum_{j=0}^{j=Y}(f(x,y))\right)$$

Where f(x,y) describes the image.

Tracking the Red Ball

The PDA, after performing the algorithms described above, has calculated numbers that represent the location of the red ball in an image frame. The next step is to control the robot to intercept the ball. The TJ Pro™ robot executes a program that makes it a slave to serial port input. The PDA has control of the motor speeds and can read robot sensors through the robot's serial port. As the PDA processes each frame, it determines the number of red pixels in the frame and whether or not the ball is even in the image. If the robot does not detect a red blob, it will slowly spin looking for the ball. If the robot sees the ball, the robot uses the centroid to guide it toward the ball. If the ball is in the center of the frame, plus or minus an experimentally defined amount, the robot goes forward. When the centroid moves toward the edges of the frame the PDA instructs the robot to move right or left to center the ball.

User Interface

The PDA GUI offers a significant advantage over a dedicated processor interface. The PDA screen can display real-time, processed images, enabling the programmer to see through the eyes of the robot. Figure 2 shows images of the screen while the PDA is processing the image while Figure 3 shows the raw image right out of the camera.

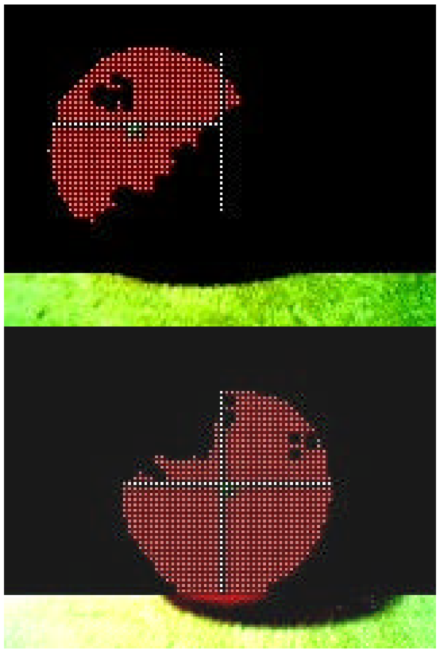

Figure 2: Screen shots of the robot tracking the red ball

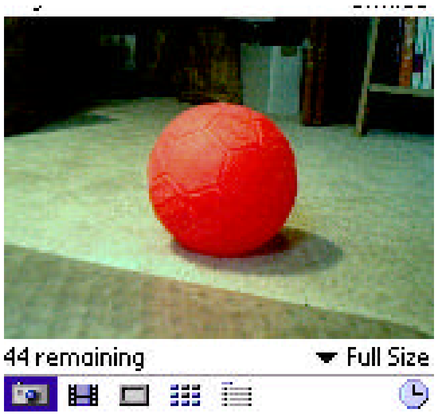

Figure 3: What the unprocessed image looks like through the eyemodule

The processed image shows the red ball as perfectly red and all other non-red pixels as black. The white lines are not just cross hairs they are the data used for the centroid calculation shown in visual representation. The small yellow/green dot is the centroid as calculated.

Algorithm performance

Our research demonstrates that a PDA can be used for a vision system on a small, low cost autonomous robot. We demonstrate that common centroids and blob detection algorithms, mainstays of vision research are efficient and applicable to a small robot with limited processing power. Our experiments with color algorithms yielded some surprises. We had assumed that HSV space would be the best for color determination, but too computationally intensive to use in its full form. In the search for simplification, we discovered that the YUV space transform yielded an extremely efficient algorithm that actually performed better in differentiating colors than the HSV transform. In our tests the HSV algorithm varied between 2.5 frames per second (fps) and 3.7 fps and our custom algorithm stayed at 4.76 fps. Since our algorithm does not change with color, unlike the optimized HSV algorithm, it not only executes faster but is easier to work with. In real world robotics having the frame rate change as the image changes could be a problem. Eliminating this problem through more detailed HSV calculations reduces performance to 2.5 fps, or about half the speed of our algorithm. Even with the improvement realized here, there is much yet to do in the area of color determination on mobile vision-equipped robots.

PDAs as vision systems

The *HandSpring* PDA and *Eyemodule* camera made a great combination for vision work. The size of the PDA and camera were perfect for our TJ Pro™ robots. The GNU and *PalmOS* development environment are not as sophisticated as Visual C++ for windows but certainly much better than the development tools for a custom micro controller board. The speed limitations were less of a problem than the memory constraints. The *PalmOS* cannot easily deal with memory blocks greater than 32K in either a static mode or in dynamic memory allocation. The *HandSpring* speed may easily be increased to 54Mhz by using an overclocking program. Some experimentation would be needed to find a working baud rate at this speed since the serial clock would also be skewed. The cost of the system was a main advantage being less than half the cost of a low level laptop or desktop computer

Future Work

We look forward to newer generations of PDAs and ultra small laptops giving our robots even greater speed. More research in color differentiation is needed, possibility using neural networks for color determination. Not covered in this paper, but of possibly great practical value, is the use of the on-board DSP in the *Eyemodule* to adjust the color of the image, making the color processing easier and faster.

References

[1] Bandlow T. et al, *Fast Image Segmentation, Object Recognition and Localisation in a RoboCup Scenari*o, Lecture Notes in Artificial Intelligence Volume 1856, Springer, Berlin, 2000.

[2] Bartelds, R. et al, *Clockwork Orange: The Dutch RoboSoccer Tea*m, RoboCup 2001, Springer-Verlag, Berlin, 2001.

[3] Berry A., *Soccer Robots with Local Visio*n, Honours [sic] Dissertation, Univ. of Western Australia, Dept. of Electrical and Electronic Engineering, 1999.

[4] Carlson, N. R. *Physiology of Behavior* (5th ed.). Needham Heights, MA: Allyn and Bacon, 1994.

[5] Fairhurst, M. C. *Computer Vision for Robotic Systems: an Introduction.* New York: Prentice Hall, 1988.

[6]Foley, J. D., van Dam, A., Feiner S. K., Hughes, J. F. *Computer Graphics Principles and Practice*. New York: Addison-Wesley, 1997.

[7] Foster, L. *Palm OS Programming Bible*. Foster City, CA: IDG books, 2000.

[8] Fu, K. S., Gonzalez, R. C., & Lee, C. S. G. *Robotics: Control, Sensing, Vision and Intelligence.* New York: McGraw Hill, 1987.

[9] Gonzalez R. and Woods R., *Digital Image Processin*g, Addison-Wesley, Reading
Massachusetts, 1992.

[10] Mead, C. *Analog VLSI and Neural Systems.* Reading, MA: Addison-Wesley, 1989..

[11] Pratt W., *Digital Image Processing, Second Edition*, John Wiley & Sons, New York,
1991.

Reiser, M.. *PC-eye Bot / Toro.* [On-line]. Available: www.mil.ufl.edu/imdl 1999.