Linux Robot with Omnidirectional Vision

George Francis **Libor Spacek**
Department of Computer Science
University of Essex
Wivenhoe Park
Colchester, CO4 3SQ
{gwfran, spacl}@essex.ac.uk

Abstract

Described here is an integrated miniaturised robot with an omnidirectional visual sensor. The hardware consists of a Gumstix computer and an image sensor mounted on a small fast autonomous mobile robot. An experiment follows which demonstrates that a simple omnidirectional vision method produces reliable wall avoiding behaviour in the robot.

This work is part of an on-going research project to explore visual guidance of small robots. The results presented here involve a catadioptric system using a low-resolution camera and a spherical mirror. Visual guidance of the robot is currently similar to that of a Braitenberg Vehicle. The robot moves freely within a designated area marked by a brighter floor covering.

In the next stage of the project, a higher resolution camera will be used with a conical shaped mirror and more complex vision algorithms. Use will be made of biologically inspired optic flow techniques to achieve real-time obstacle avoidance.

1. Introduction

Autonomous robot guidance can be decomposed into two main aspects: collision avoidance, and navigation towards a goal. There is a difference in the time requirements of these two aspects; collision avoidance is an immediate safety concern for the robot, while the goal seeking can take place over a longer timescale (Gaspar et al., 2000).

The choice of sensors is limited for small, low-cost, battery powered utility robots operating at reasonable speeds in an un-structured (human) environment. Sonars have a slow update rate, Laser Range-Finders are large and expensive, and Infra-red distance sensors have very limited coverage. Vision has its own problems but the raw sensing is fast and a catadioptric omnidirectional sensor can detect most hazards to a small robot.

With the immediate safety of the robot secure, the remaining processing power on the robot can be used to perform navigation (at a slower rate) using methods such as those based on landmarks (Gaspar et al., 2000) or on the appearance of the horizon (Rushant and Spacek, 2004).

However, all of the above depends on being able to process images in real-time. Until recently, computers capable of doing this have been relatively bulky and power-hungry.

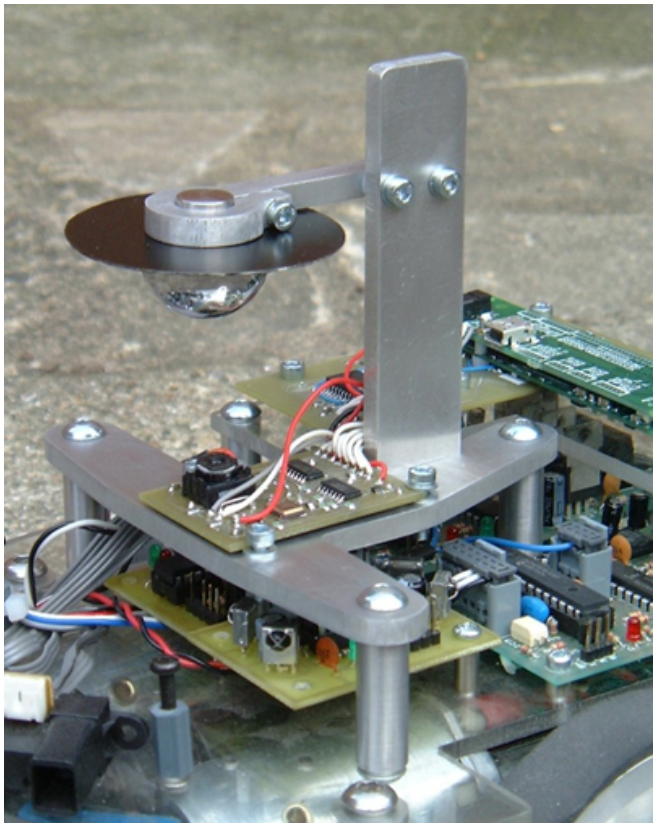

Figure 1: The Prototype Omnidirectional Camera.

The consumer electronics industry is rapidly improving the design of small, high-performance, low-power computing devices such as Mobile Phones and Personal Digital Assistants. This technology is steadily being migrated to embedded industrial control applications. Re-

cent advances have made available very small general purpose computers with low power consumption and yet enough processing capability to perform simple vision processing in real-time.

Small footprint Single Board Computers (SBCs) are available from a range of companies. One such company, Gumstix Inc in the USA, is manufacturing a particularly small but still fully featured SBC that runs the popular Linux operating system.

Such a small size computer raises the interesting possibility of mounting the camera and computer together in the same module to produce an integrated vision module. This has been done before in the shape of the popular CMUcam2 (Rowe, 2003). This is a micro-controller/camera module that performs colour blob tracking and very low resolution (8x8 pixel) frame differencing in real-time and can output a steering signal via a serial link or drive model radio control servos directly.

The main limitations of this device are the lack of processing power in the micro-controller, and lack of a full resolution frame buffer. By using a Gumstix computer instead of a micro-controller, much more powerful vision processing has been achieved.

Using a full-featured operating system such as Linux on a small embedded system presents a host of advantages during development. One example is the use of Network File System (NFS) to access the directories of a remote host via the USB port, thus simplifying file transfer and cross-compiling.

The Gumstix computer doesn't have a port to take video input directly, so an interface had to be designed and built to connect the image sensor to the CPU bus (which is accessible via a 92-pin connector).

Having built the interface and obtained images on the Gumstix, attention turned to using the system to guide a small mobile robot.

There are two obvious ways of using a single camera for visual guidance: either point it forwards, or use a rotationally symmetrical mirror to create an omnidirectional system.

A forward-looking camera will have a narrow field of view. The camera used here is quoted as 50^o in the horizontal plane by the manufacturer, (STMicroelectronics, 2004). This means it will have difficulty detecting objects close to the side of the robot. For this reason, an omnidirectional system, in this case having a 180^o field of view, was deemed to be a better choice.

For speed of implementation, the decision was made to perform very simple visual guidance of the robot.

The robot has been programmed to avoid dark objects on a plain light-coloured floor using the principles of a Braitenberg Vehicle (Braitenberg, 1984).

By enclosing an area of floor with a darker coloured material (floor marking), the robot executes the wall-avoiding behavior.

This task has traditionally been performed using sonar or infra-red distance measuring sensors and real walls, with the attendant range and interference problems associated with active sensors, and the resulting danger of crashing into the walls.

An active area of research is simultaneous location and mapping with single cameras (Davison, 2003). While the computational overhead of Davison's method is probably too high for the processor used here, the use of active markers (Cassinis et al., 2005) would remove the computation of matching features in the image with previously stored templates. (Davison et al., 2004) find that using wider angle lenses improves their technique, so the further extension is to use omnidirectional vision. More of the active markers are visible at any one time when using an omnidirectional camera, which improves the accuracy and reliability of such a system.

2. Hardware Design

2.1 Camera Interface Circuit

The camera is based around the VS6502 320x240 pixel colour sensor (STMicroelectronics, 2004). Fig 2 shows the major components of the camera interface to the Gumstix (connections to the Gumstix are at the bottom of the diagram).

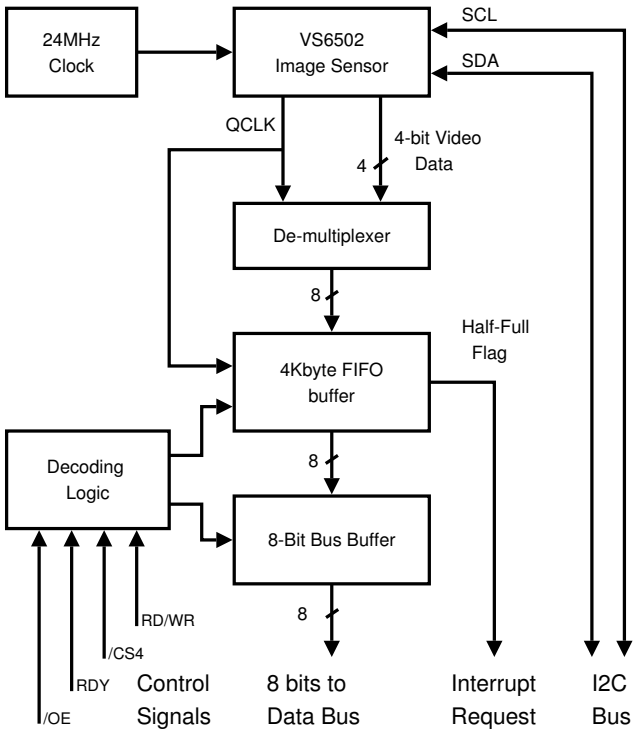

Figure 2: Block Diagram of Camera Interface circuit.

The camera sensor outputs data at a fixed high rate

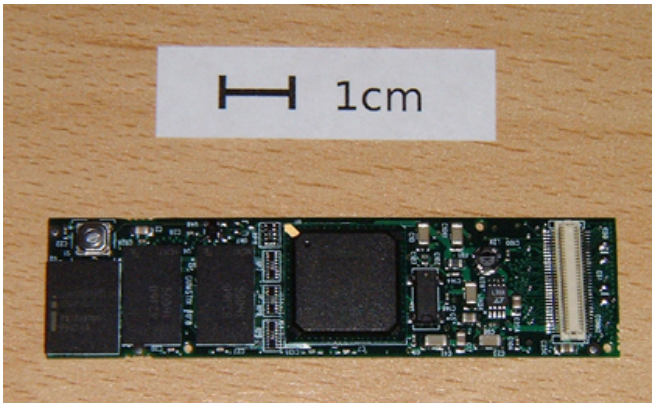

Figure 3: Gumstix Linux Computer. (400MHz ARM Processor with 64Mb RAM and 4MB Flash)

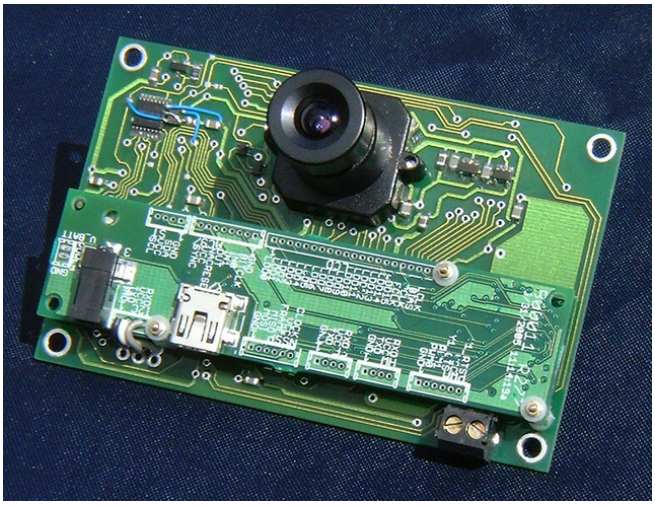

Figure 4: The new integrated vision module (based on the Kodak KAC-9618 image sensor). Overall Size: 83 x 54 x 30mm

which is determined by the clock frequency and the configuration of the internal control registers. These registers control exposure, frame rate, etc. and are accessed via a separate serial link (an I^2C bus).

Due to the continuous high speed data flow, problems may occur if the computer is not ready to receive data (eg delays due to interrupt priorities and latency). For this reason, a Dual Port First-In/First-Out (FIFO) buffer is used to store data until the computer operating system is in a position to handle it. Data is clocked into the buffer at a fixed rate and then read out with high-speed bursts as and when the computer is ready. The FIFO buffer has a half-full output flag which is used to trigger an interrupt request to the computer. Before being fed to the FIFO buffer, the 4-bit video data (which consists of high and low nibbles sent consecutively) is converted to 8-bit form.

This camera is a prototype and is implemented on two

Figure 5: Test Arena. The area consists of a sheet of white vinyl approximately 1.5m by 1.25m. It is shown here resting on a darker sheet to improve the contrast of it's edge against the light coloured concrete.

small boards. In the future, the interface can be implemented on a single printed circuit board which the gumstix computer will clip onto. A Small expansion board (available from Gumstix Inc) will clip onto the other side of the computer to give access to the I^2C bus (to configure the sensor), Serial Port (to control the robot) and USB Port (for communication with a PC).

2.2 Embedded Linux Computer

The computer used in this project is a Gumstix Connex 400 (see Fig 3), full details of which are available on the company's website (www.gumstix.org). It measures just 80mm long by 20mm wide. The processor is an Intel XScale PXA255 400MHz (an ARM-5 processor), the board has 4Mb of flash memory, 64Mb of DRAM, and runs the Linux operating system (kernel version 2.6). Communication with a host PC can be either RS232 Serial or USB. It has a 92 pin bus header giving access to the cpu bus, which will be vital to interface the camera to. Power consumption is claimed to be 50mA when idle, rising to 250mA with the processor fully loaded.

The main limitation of a Gumstix computer for this application is the lack of a floating point maths unit in the processor, which may limit the quantity and type of vision processing that can be done.

2.3 Camera Device Driver

To support the new camera, a Linux device driver needed to be written. An advantage of an embedded system is the 'single use' nature of the overall system. Since the camera doesn't have to work with a wide range of different computer systems, some shortcuts can be taken

in the design of the driver which would normally render it incompatible with many computers. One convenient shortcut is restricting the amount of memory used by the kernel by passing it an argument at boot time. This means that the remaining memory can be used by the driver without having to go through the kernel memory-mapping and registration functions.

The driver has to perform two functions. The first is to setup the control registers in the camera sensor via the I^2C bus, the second is to receive the video images. Due to the large amount of data contained in an uncompressed video stream, the processor's built-in Direct Memory Access (DMA) hardware is used to transfer the data from the camera to the processor memory. The interrupt request (triggered by the FIFO half-full flag, see above) can be used to start a DMA transfer to copy a large block of data (in this case 2K bytes) from the FIFO buffer to system RAM. Using the hardware DMA module leaves the cpu free to process the images in real-time. The ARM Processor used in the gumstix also supports a fast interrupt request architecture called FIQ (Furber, 2000), which will be used to reduce the interrupt latency. A faster interrupt routine also reduces the cpu workload allowing more image processing to be done.

2.4 Mobile Robot

The robot used here is a simple small battery powered differential drive robot that was built as a prototype for a previous project. It has a solid aluminium chassis and machined aluminium wheels, is driven by 2 x 12V 9W Bosch geared motors with right-angle drive, and is powered by 12V 2100mAh Ni-Cad Batteries. The motors are controlled by H-bridge speed controllers (each with a microprocessor running open-loop PWM voltage control). Approximate robot size: 280mm long x 240mm wide x 120mm high.

The robot has a simple serial control interface using an I^2C bus. The only control available is bi-directional proportional voltage output to the two motors.

2.5 Catadioptric Sensor

Many of the radially curved mirrors typically used for catadioptric sensors (parabolic, hyperbolic etc) are difficult and expensive to make. Conical mirrors have many advantages (Spacek, 2005) and can be machined on a simple lathe. Spherical reflecting objects are readily available in the shape of stainless steel ball bearings, and give a very wide angle (though distorted projection and low resolution) field of view (see Figure 6).

Though conical mirrors give better resolution images, the vertical field of view is only half the field of view of the camera, so a wide-angle camera lens would need to be used. The prototype camera has a fixed lens with a 40^o field of view in the vertical plane. Using a conical mirror with this lens would give an annular image covering just 20^o vertically, resulting in the robot not being able to see objects on the ground close to it. For this reason, a spherical mirror was chosen for the first stage of the project reported here. A conical mirror will be used with the new camera with an interchangeable lens (see Fig 4).

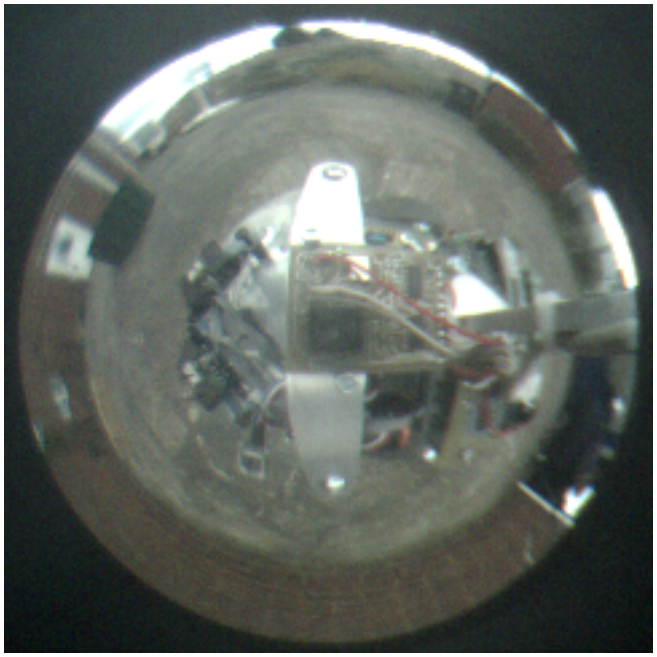

Figure 6: An example image obtained from the prototype omnidirectional camera (contrast-stretched to enhance printing). The front of the robot is facing towards the left of the picture.

The lack of a single viewpoint with the spherical mirror precludes accurate un-warping to a perspective projection image. However, this is not necessary with the prototype camera as the simple vision algorithms are applied directly to the original un-warped images.

3. Visual Guidance Software

The first visual guidance software implemented on the camera system is simply to turn away from dark objects and slow down if approaching too closely. To detect dark objects to the front and sides of the robot, a threshold is applied to pixels in the image that correspond to positions around the robot. The threshold used is selected automatically, so the detection algorithm works on any lighter/darker combination of floor colours.

Throughout this work, the floor is assumed to be flat and the positions of pixels in the image are assumed to be a direct mapping for points on the floor. Mechanical alignment is relied upon to ensure the view of the mirror is in the correct position in the image. There is no auto detection of the mirror in the image at this stage.

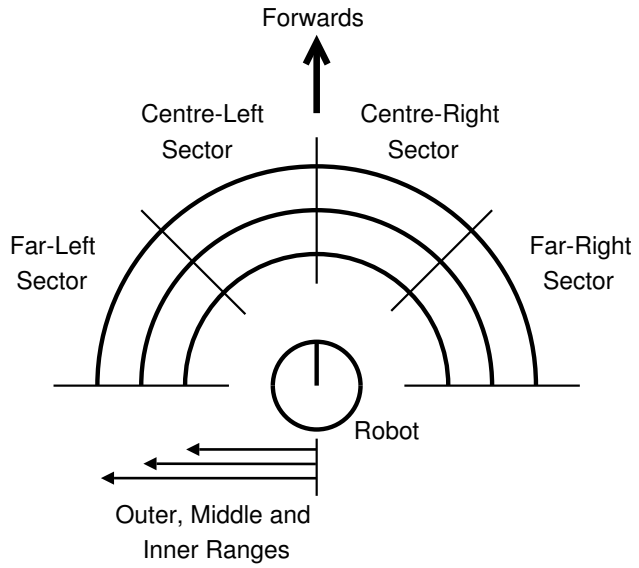

Figure 7: Detection Arcs. Only pixels along the three semi-circles are considered in the image.

Positions equidistant from the robot map to circles in the image. By selecting a radius (corresponding to the desired distance from the robot) and examining the grey-level of pixels along that circle, simple obstacle detection can be implemented. As the robot only travels forwards in this experiment, only the forward 180^o of the image is considered.

The object detector function reads pixels that lie along arcs drawn in front of the robot and builds a histogram of their grey-levels. A threshold is selected by looking for the first minima (after the first maxima) in the histogram, and flags are then set in an array for those sectors that contain pixel values below that threshold. These flags are used to generate the speed and steering signals to send to the robot controller.

An easy method for calculating the 2D positions of pixels that lie on a circle is Bresenham's Midpoint algorithm. This generates points for an arc that is $1/8^{th}$ of a circle, the rest of the circle can be drawn by symmetry. This led naturally to the idea of four sectors in front of the robot (four of the eight arcs, see Fig 7). Three different radii have been used to give an indication of the distance to the object.

The result of the detector function is a 4 x 3 array containing a 1 for each segment that an object was detected in (ie contained pixels with grey-levels below the threshold) and zeros for the rest. So for each of the four sectors (Far-Left, Centre-Left, Centre-Right and Far-Right) there is an indication of whether an object has been detected, and how far away it is (Outer-Range, Middle-Range and Inner Range).

The robot steering signal is then calculated as follows: If an object is in a far sector, turn slowly away from it; if it is in a centre sector, turn quickly away from it.

Similarly, for the speed signal: If there are no objects, or an object is at the outer range, drive at normal speed; If an object is at the middle range, drive at slower speed; And if an object is at the inner range, stop (so that the robot turns on the spot).

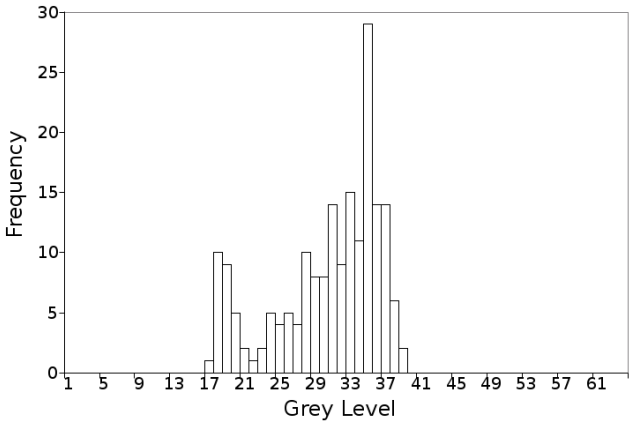

Figure 8: Histogram of a low-contrast image (the dark object shown in the upper-left of Fig 6). Threshold selection is more difficult here due to the overlap between the grey-level populations of the background and the object

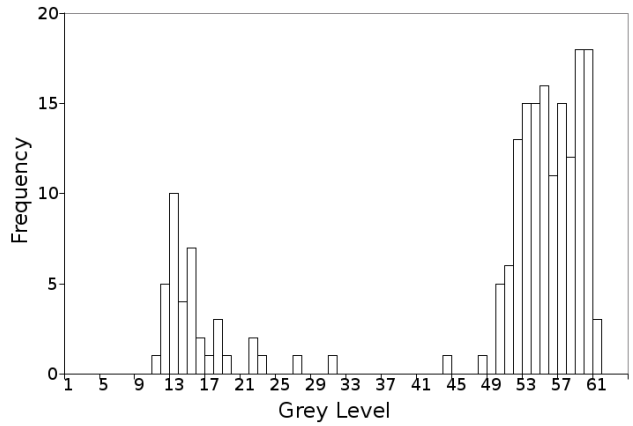

Figure 9: Histogram of pixels along the inner arc of an image from the edge of the high-contrast arena (image in Fig 11).

3.1 Threshold selection

The threshold for object detection is calculated on a frame-by-frame basis using an automatic thresholding approach. The first minima in the histogram is detected as follows: A histogram is constructed by reading the pixel values along the current arc. The histogram is then stepped through one grey-level at a time (starting from the black-level), and a record is kept of the maximum histogram value so far seen. When the histogram value falls to less than half of the recorded maximum, the grey-level currently being checked in the histogram is used as

the threshold. If there are no objects in view, then the majority of the pixels will fall into one group. To check for this, a cumulative total of the histogram is also kept, and if the first minima occurs after the cumulative total has risen past 2/3 of the total number of pixels in the histogram (ie most of the pixels are in the first maxima) then a threshold of zero is used. So if most of the pixels are in the first group (as would be the case if there were no objects and all pixels are background) then using zero for the threshold ensures the detector output is also zero.

This has the obvious problem that if the robot is viewing mostly object (ie it has moved too close to an inside corner) then the algorithm will fail to detect the object at all. However, the steering calculation is prioritized according to distance, the inner arcs taking precedence over the outer arcs. This means that as long as at least the inner arc is viewing mostly background, the robot can cope with getting close to a corner.

Figures 8 & 9 show two example histograms recorded from the robot. The first shows the difficulty of performing histogram analysis on histograms with so few points. Random variation in pixel values gives very noisy histograms. This could be improved by increasing the number of pixels considered. Alternatively, highly contrasting colours can be used to separate the grey-level populations for the background and the arena (Fig 9).

One of the first problems encountered in threshold selection was, as always, with the illumination of the images. Automatic exposure control software had to be written, the approach taken was to adjust the exposure so that the peak of the histogram for the inner arc fell close to a pre-set value. This occasionally fails if the robot has moved too close to a large object, so that there is more dark object in the histogram than light background.

To overcome this problem, a known colour patch was fitted to the robot. An area of pixels within the image of this patch was then used to calculate the exposure (instead of the inner arc), resulting in more robust performance. A disadvantage of many radially curved mirrors is that the image of the robot occupies a considerable proportion of the total image area. Here we are turning this to our advantage.

Another problem is un-even illumination across the image. An omnidirectional camera observes a large area of ground in each frame, and occasionally minima appeared in the histogram due to un-even illumination rather than objects. This could only be resolved by taking the local brightness of an area of the image, rather than averaging it over the whole image.

4. Experiment

Images are obtained for illustration purposes by having the image processing software write the image buffer out to a laptop (over NFS via the USB connection). Two

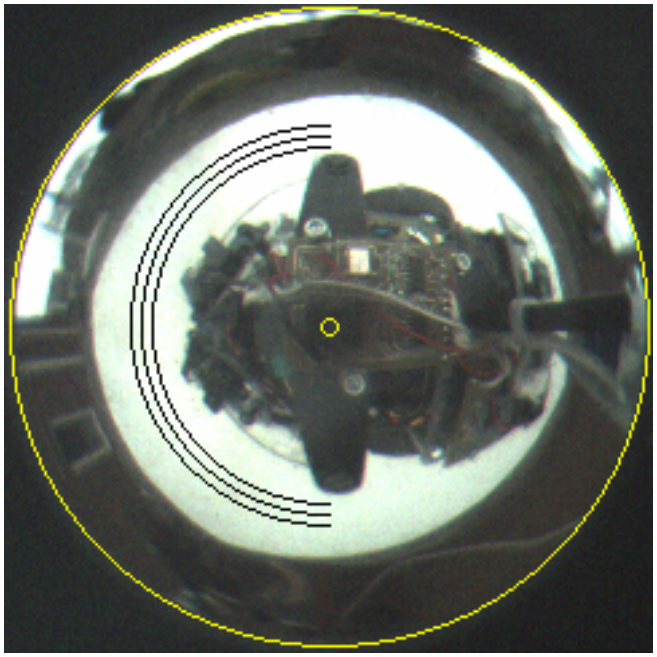

Figure 10: Image taken in the middle of the arena. The robot is facing to the left of the image. The three black arcs (in front of the robot) indicate that no objects are currently being detected.

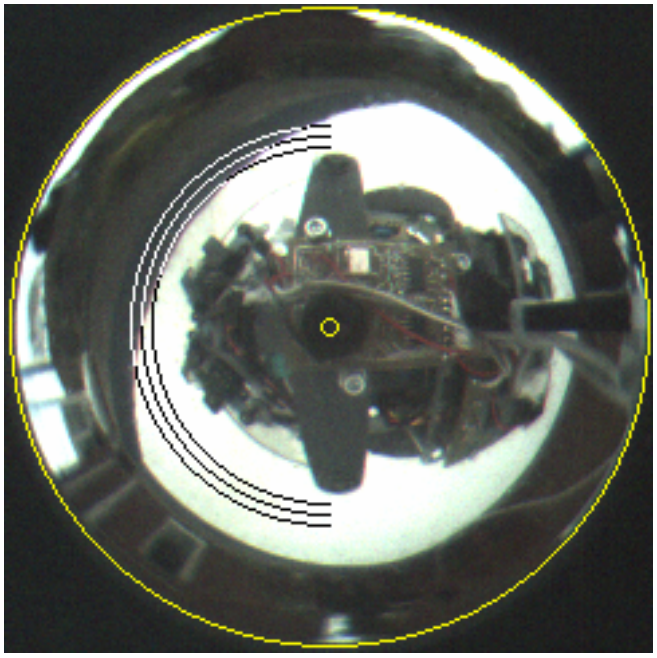

Figure 11: Image taken at the edge of the arena. The robot is facing to the left of the image. Parts of the arcs have now been coloured white by the image-processing software, indicating that the grey-level of these pixels is below the threshold thus denoting an object to avoid.

examples of these are shown in Figures 10 & 11. Arcs have been added by the image processing software to show the behaviour of the object detection algorithm.

Also visible are the outer circles which are used to mechanically align the camera and mirror. The mirror position was adjusted mechanically until its image fell within the outer circle. Parts of the aluminium mounting frame had to be painted black to reduce the reflection of sunlight into the lens.

The opportunity was also taken to measure the current consumption of the Gumstix/Camera module while it was processing images. The supply voltage was $5.0V$ and the supply current was measured as $192mA$.

4.1 Behaviour and Results

By programming the robot to turn away from dark objects and then placing it within a light-coloured area on a darker floor, an emulation of the traditional wall-avoiding behavior has been achieved. The dark part of the floor acts as a virtual wall; every time the robot approaches it, it turns away. The robot is set to drive straight forwards when not detecting dark floor, so it continually drives towards an edge of the light area, turns away from it, and drives on towards another edge.

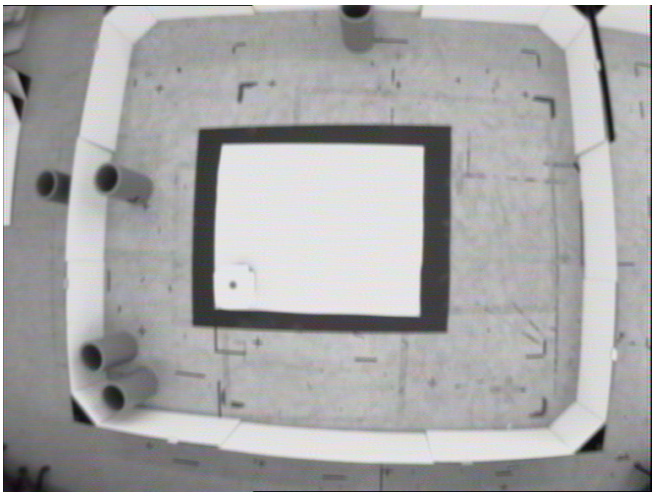

Figure 12: View of the test arena (inner white rectangle) from the overhead logging camera. The tracking-target (completely covering the top of the robot) is visible in the lower-left corner.

To record the behaviour of the robot, a conventional overhead camera logging system has been used. This uses template matching to track and log the pixel position of a visual target. The logging system also records an image of the arena at the beginning of each recording run. The tracking data subsequently recorded has then been plotted to the same scale as the image, and superimposed over it. In this way, the path of the robot can be seen in relation to the edges of the arena. Figure 12 shows the wide-angle view from the overhead camera, and Figure 13 shows the data from approximately 150 seconds of robot running superimposed on an enlargement of the wide-angle view.

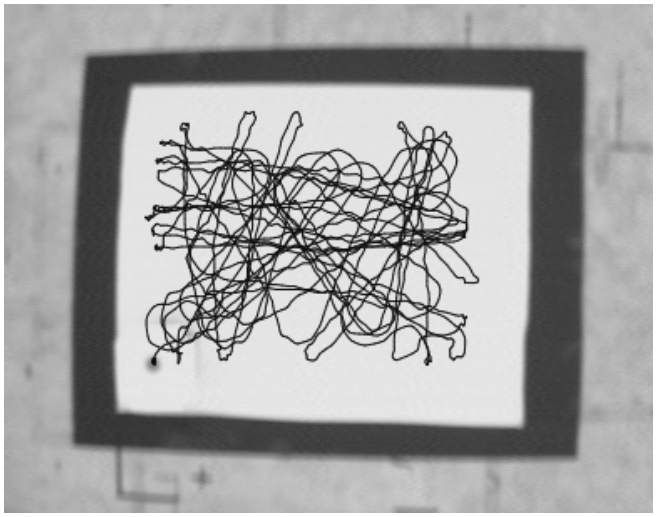

Figure 13: Test run of the robot (showing approx $2\frac{1}{2}$ minutes of data). Data from the overhead logging camera has been superimposed over the reference image recorded by the camera. An enlargement showing the arena is shown here.

As can be seen from Figure 13, the angle by which the robot turns depends on the angle of approach to an edge. The robot turns away less when it approaches the edge at a shallow angle. Conversely, if the robot is heading directly towards the edge, it gets closer to that edge before turning back the way it came. This is caused by the time it takes to turn coupled with the forward speed of the robot. When the turn is initiated, the object is still getting closer, and the robot often gets close enough to the edge for the software to stop the forward motion before the turn is complete (the software slows and then stops the robot as the range to the object progressively decreases). In this experiment, the robot's occasional slowing to a stop while turning was caused by the choice of parameters (speed and rate of turn), rather than by a problem in being able to detect the boundary.

The very wide angle view provided by Omnidirectional Vision has proved very useful, particularly in situations where the robot has to get out of a corner. In these cases, it continues to turn until it is facing open space before driving away. Boundaries can be seen from some distance enabling the robot to react smoothly.

A short video of the robot demonstrating this behaviour is available at this address: `http://cswww.essex.ac.uk/mv/omnipapers/LinuxRob.mpg`

5. Conclusions

This paper has successfully demonstrated simple visual processing in real-time on a small mobile robot.

The choice of modern hardware components has produced a robust low-power vision system. That system has been combined with a spherical mirror to create an omnidirectional camera with simple real-time vision processing. The use of a tiny computer board with a full-featured operating system, not normally found on robots as small as this one, has greatly eased system development.

The wide angle of view provided by the use of omnidirectional vision has been very useful. The edges of the arena are detected easily from a distance and are not lost as the robot turns, enabling it to escape from corners reliably. Despite the low resolution of the camera, the edges of the arena are still detected at a distance of 1m (eight times the robot radius).

6. Future Work

Our new camera has now been completed (see Fig 4) which will result in improved resolution, faster frame rate, and will realize the goal of developing a small standalone vision module that can easily be added to any mobile robot. The limits of this type of processor for vision applications will be investigated.

Cone mirrors have many advantages as discussed by (Spacek, 2004, Spacek, 2005). Use of such a mirror on the new camera is expected to improve performance still further due to the area of interest in omnidirectional vision being spread over a larger area of the sensor.

Implementing more complex algorithms such as Omni Flow (Spacek, 2004) should also be possible in view of the capability of the processor used here. By using optic flow techniques based on the behaviour of insects (Webb et al., 2004), obstacle avoidance is possible using considerably less processing than conventional techniques.

References

Braitenberg, V. (1984). *Vehicles: Experiments in synthetic psychology.* MIT press.

Cassinis, R., Tampalini, F., and Fedrigotti, F. (2005). Active markers for outdoor and indoor robot localization. In *Proceedings of TAROS 2005, Towards Autonomous Robotics Systems.*

Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the ninth international Conference on Computer Vision ICCV'03, Nice, France, October 2003. IEEE Computer Society Press.*

Davison, A. J., Cid, Y. G., and Kita, N. (2004). Real-time 3d slam with wide-angle vision. In *IAV2004 - Preprints 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles, Instituto Superior Tcnico, Lisboa, Portugal, July 5-7, 2004.*

Furber, S. (2000). *ARM system-on-chip architecture (second edition).* Addison-Wesley.

Gaspar, J., Winters, N., and Santos-Victor, J. (2000). Vision-based navigation and environmental representations with an omnidirectional camera. In *IEEE Transactions on Robotics and Automation, 16(6):890–898, 2000.*

Rowe, A. (2003). *CMUcam User Guide.* `http://www.seattlerobotics.com/CMUcam2_manual.pdf`.

Rushant, K. and Spacek, L. (2004). An autonomous vehicle navigation system using panoramic machine vision techniques. In *ISIRS98, Bangalore India, pp. 275-282, January 1998.*

Spacek, L. (2004). Coaxial omnidirectional stereopsis. In *ECCV (4)*, pages 354–365.

Spacek, L. (2004). Omni Flow. In *Proceedings of TAROS 2004, Towards Autonomous Robotics Systems.*

Spacek, L. (2005). A catadioptric sensor with multiple viewpoints. *Robotics and Autonomous Systems*, 51(1):3–15.

STMicroelectronics (2004). *VS6502 VGA Color CMOS Image Sensor Module.* `http://www.st.com/stonline/products/literature/ds/10659.pdf`.

Webb, B., Harrison, R. R., and Willis, M. A. (2004). Sensorimotor control of navigation in arthropod and artificial systems. *Arthropod Structure & Development*, 33:301–329.