VISION-BASED OBSTACLE AVOIDANCE FOR A SMALL, LOW-COST ROBOT

Chau Nguyen Viet, Ian Marshall
Computer Science Department, University of Kent, Canterbury, United Kingdom
cn41@kent.ac.uk, ian.marshall@kent.ac.uk

Keywords: obstacle-avoidance, robot vision.

Abstract: This paper presents a vision-based obstacle avoidance algorithm for a small indoor mobile robot built from low-cost, and off-the-shelf electronics. The obstacle avoidance problem in robotics has been researched extensively and there are many well established algorithms for this problem. However, most of these algorithms are developed for large robots with expensive, specialised sensors, and powerful computing platforms. We have developed an algorithm that can be implemented on very small robots with low-cost electronics and small computing platforms. Our vision-based obstacle detection algorithm is fast and works with very low resolution images. The control mechanism utilises both visual information and sonar sensor's measurement without having to fuse the data into a model or common representation. The robot platform was tested in an unstructured office environment and demonstrated a reliable obstacle avoidance behaviour.

1 INTRODUCTION

Autonomous navigation in an unstructured environment, i.e. an environment that is not modified specifically to suit the robot, is a very challenging task. Current robots that can operate autonomously in an unmodified environment are often large and expensive. Most of today robot navigation algorithms rely on heavy and power-hungry sensors such as laser range finders, high resolution stereo-visions (Thrun et al., 1999; Manduchi et al., 2005; Stentz et al., 2003; Ibanez-Guzman et al., 2004; Batalin et al., 2003). As a consequence, these robots require powerful computing units to be mounted on-board. The size, computational power, and energy requirements of these robots limit the range of their applications and operational period. In this work, we have built a small mobile robot from cheap off-the-self electronics to perform obstacle avoidance in an unstructured environment. Obstacle avoidance is the first basic behaviour needed for an autonomous mobile robot. The robot is equipped with a low-power camera and two ultrasonic sensors. Image processing is done in real-time and on-board. Our robot is small and energy efficient; it is powered by AA batteries.

Obstacle avoidance is one of the most fundamental and researched problems in the field of mobile robotics. Most obstacle avoidance algorithms uses active range sensors such as ultrasonic sensors, laser range finders and infra-red sensors. Visual sensors are an alternative solution for obstacle avoidance and becoming increasingly popular in robotics. Visual sensors often provides better resolution data, longer ranges at faster rates than range sensors. Because visual sensors are inactive they are less dependent on the environment. However image processing is a very computationally expensive task. Vision often require complicated software and powerful computing platform or dedicated hardware module. For very small robots, i.e. those that are man-carriable, vision is still rare.

To perform obstacle avoidance, a robot needs to know the distances to objects around it. The most common method of extracting depth information from visual images is stereo-vision. Stereo vision often produces accurate depth maps. The downside is this approach requires powerful computation platform, complex calibration process, and two cameras. For small constrained robot platforms, stereo-vision is not hard to implement. Depth information can be

extracted from a sequence of images from a single camera using motion parallax (Lu et al., 2004). This technique was used for obstacle avoidance in (Santos-Victor et al., 1993; Zufferey and Floreano, 2005). For the output of this algorithm to be accurate, the image processing rate must be high enough e.g. over 30 frames per second. Another class of algorithm is based on colour-based terrain segmentation (Lorigo et al., 1997; Lenser and Veloso, 2003; Ulrich and Nourbakhsh, 2000). This approach works on a single image. If we can assume the robot is operating on a flat surface and all objects have their bases on the ground, the distance from the robot to an object is linear to the y-axis coordinate of the object in the perceived image. The problem is reduced to classifying a pixel into two classes, obstacle or traversable ground. This approach is suitable for robots that operate indoor or on benign flat terrains. Since it does not requires high resolution images or high frame rates camera, we adopt this approach for our vision module. What makes our algorithm different from existing algorithms is the use of a lookup map for colour classification and a reduced colour space. Lookup map is a very fast classification method. On a Gumstix computer clocks at 200 MHz, our algorithm can process more than 500 frames of $87 * 44$ pixels per second. The vision algorithm presented in (Lenser and Veloso, 2003) uses 3 array access operations and an AND bitwise operations for each pixel. Our algorithm uses only one array access operation. Lugino and her group developed an algorithm that can work with low resolution image 64*64 pixels frame in (Lorigo et al., 1997). Our algorithm works with even use lower resolution image of 22*30 pixels frame. This reduces the computing cycle required for the vision algorithm and enable our algorithm to run on embedded computing devices.

Although our vision algorithm is reliable, due to hardware limitation, the camera we uses has a narrow field of view (FOV), two additional sonar sensors were added to expand the robot's FOV. This does not conflict with our preference of vision sensor over range sensor. Vision is the main sensor and the sonar sensors were added to improve the performance of the system only. The control mechanism is reactive, it has no memory and acts upon the most current sensor readings only. This allows the robot responds quickly to changes in the environment. While many other systems fuse data from different sensor sources into a single representation or world model, our algorithm does not. There is no attempt to fuse distance estimates from visual images and sonar sensors into any kind of model or representation. Instead, the control rules are tightly coupled with sensory data and the hardware configuration. This gives rise to a fast and robust obstacle avoidance behaviour. Our robot is able to response to changes in the environment in real-time. The approach we used is inspired by the subsumption architecture (Brooks, 1985) and Braintenberg vehicles (Braitenberg, 1984).

Our main contribution is an obstacle avoidance algorithms that uses low-power off-the-self camera and runs on a small constrained platform. Even with very low resolution colour images our robot demonstrates a robust obstacle avoidance behaviour. The robot was tested in a real office environment and was shown to be very robust; the robot could operate autonomously over a long period of time. This work emphasised the efficiency and simplicity of the system. We want to build an autonomous robot with the minimal hardware configuration. We implemented the controller on the Gumstix Linux computer, which runs at 200 MHz. But the algorithm can run on a slower microprocessor as it uses only a small fraction of the Gumstix's processing cycle. The obstacle avoidance algorithm might be used as a module in a more complex system e.g. the first level of competence in a subsumption architecture. It can be used on it own in an application such as exploration, surveillance. Because only a small fraction of the CPU is required for obstacle avoidance, more spaces are available for more complex behaviours on the platform.

This paper is organised as follows. In section II, we present the vision algorithm and control mechanism. Section III describes the hardware configuration and software implementation. The experiments and results are reported in section IV. In section V, we conclude with some remarks and our future plan.

2 Vision and Control Algorithm

2.1 Ground and obstacles segmentation

In our classification algorithm, pixels are classified according to their colour appearance only. The colour space we use is the RGB colour space. Each colour in the colour space is set to be a ground or obstacle colour. This classification information is stored in a binary lookup map. The map is implemented as a three dimensions vector of integers. To classify a pixel, its RGB components are used as indices to access the class type of the pixel. The classification process is very fast since for each pixel only one array lookup operation is needed.

The lookup map is populated from example pictures of the ground. First, the algorithm counts the

number of pixels of each colour in the example pictures. Then if the number of pixels of a colour is more than 5% of the total number of pixels in those pictures, that colour is set to be a ground colour. The 5% threshold is used to eliminate the noises in the images. Procedure 1 describes this calibration process. A lookup map is also very efficient for modification. At the moment, the calibration process is done once before the robot starts moving and the lookup map remains unchanged. We anticipate that the colour appearance of the ground and the lightning condition are likely to change if the robot operates for a long period or moves into different environments and therefore any classification technique is required to adapt to these changes. In the near future, we plan to implement an on-line auto-calibrating algorithm for the vision module. Procedure 2 describes how a pixel is classified during the robot's operation.

The main drawback of using the lookup map is memory usage inefficiency. In a constrained platform the amount of memory needed to store the full 24 bits RGB space is not available. To overcome this problem, we reduce the original 24 bits RGB colour space to 12 bits and decrease the size of lookup table from 2^{24} elements to 2^{12} elements. Effectively, we lower the resolution of the colour space but also make the classifier more general since each element of the reduced table represents a group of similar colours in the original space. We also use very low resolution images of $22 * 30$ pixles. Fig. 1 has two examples of the outputs from this segmentation procedure. At the top row, there is a picture taken from the camera mounted on our robot at maximum resolution and the binary image produced by the segmentation procedure. The binary image contains some noise and a falsely classifies a part of a box as ground since their colours are similar. Nevertheless in this case, the distance to the box's base is still correctly measured. At the bottom row is the down-sampling version of the top row picture and its corresponding binary image.

The output of the image segmentation is a binary image differentiating obstacles from the ground. Assuming all objects have their bases on the ground, the distance to an object is the distance to its base. This distance is linear to the y-coordinate of the edge between the object and the floor in the binary image. For obstacle avoidance, we only need the distance and width of obstacles but not their height and depth. Therefore a vector of distances to the nearest obstacles is sufficient, we call this obstacle distance vector (ODV). We convert the binary image to the required vector by copying the lowest y-coordinate of a non-floor pixel in each column to the corresponding cell in the vector. Each element of the vector represents the distance to the nearest obstacle in a specific direction.

Procedure 1 PopulateLookupMap (n: number of pixels , P : array of n pixels

for $i = 0$ to $n-1$ **do**
  $(R,G,B) = \Leftarrow rescale(P[i]_r, P[i]_g, P[i]_b)$
  $pixel\_counter[R][G][B] \Leftarrow$
  $pixel\_counter[R][G][B] + 1$
end for
for (R,G,B) = (0,0,0) to MAX(R,G,B) **do**
  $is\_ground\_map[R][G][B] \Leftarrow$
  $pixel\_counter[R][G][B] > n * 5\%$
end for
return *is_ground_map*

Procedure 2 is_ground(p : pixel)

$(R,G,B) \Leftarrow rescale(p_r, p_g, p_b)$
return is_ground_map[R][G][B];

2.2 Control algorithm

The control algorithm we adopted is reactive. Decisions are made upon the most recent sensory readings. The inputs to the controller are the obstacle distance vector, produced by the visual module, and distance measurements from two sonar sensors pointing at the sides of the robot. The ODV gives a good resolution distance map of any obstacle in front of the robot. Each cell in the vector is the distance to the nearest obstacle in a direction of an angle of about 2.5°. The angular resolution of the two sonar sensors are much lower. So the robot has a good resolution view at the front and lower at the sides. The controlling mechanism consists of several reflexive rules.

- If there are no obstacles detected in the area monitored by the camera, run at maximum speed.
- If there are objects in front but further than the trigger distance, slow down.
- If there are objects within the trigger distance, start to turn to an open space.
- If a sonar sensor reports a very close object, within 5 *cm*, turn to the opposite direction.

The control algorithm does not calculate how far the robot should turn. It will keep turning until the area in front is clear. The robot looks for an open space by first looking in the opposite direction to the perceived obstacle, if the half image in that side is free of obstacles, the robot will turn to this direction. If

A

B

C

D

Figure 1: The images processed by the vision module. A is a full resolution colour image. B is the binary obstacle-ground image of A. C is a low resolution image from A and D is its corresponding binary image

there are obstacles in both left and right half of the image, the two measurements from sonar sensors are compared and the robot will turn to the direction of the sonar sensor that reports no existence of obstacles or a biger distance measurement. There is no attempt to incorporate or fuse data from the camera and sonar sensors together into a uniformed representation. The algorithm uses the sensor readings as they are.

3 Platform Configuration And Implementation

The robot control software runs on the Gumstix (gum,), a small Linux computer that has an Intel 200 MHz ARM processor with 64 Mb of RAM. The vision sensor is a CMUcam2 module connected to the Gumstix via a RS232 link. A Brainstem microcontroller is used to control sonar sensors and servos. The robot is driven by two servos. These electronic devices are mounted on a small three wheeled robot chassis. The total cost of all the components is less than 300 US dollars. The robot can turn on the spot with a small radius of about 5 cm. Its maximum speed is $15cm/s$. The robot is powered by 12 AA batteries. A fully charged set of batteries can last for up to 4 hours. Fig. 2 shows the area in front of the robot that is monitored by the robot's sensors. The CMUcam is mounted on the robot pointing forward at horizontal level and captures an area of about $75cm^2$. Because the camera has a relatively narrow FOV of about 55°, the two sonar sensors on the side are needed. In total, the robot's angle of view is 120°. The robots dimensions are $20cm * 20cm * 15cm$. The hardware configuration was determined by the trial and error method. The parameters we presented here were the best that we found and were used in the experiments reported in section IV.

The maximum frame resolution of the CMUCam2 is $87 * 144$ pixels, we lower the resolution to only $22 * 30$ pixels. We only need the first bottom half of the picture so the final image has dimensions of $22 * 15$. The resolution down-sampling and picture cropping is done by the CMUcam module, only the final images are sent to the control algorithm running on the Gumstix.

In our implementation, the obstacle distance vector has 22 cells, each cell corresponds to an angle of 2.5° . The vector is then outputted to the controller. Since the control algorithm doesn't build a model, there is no need to convert the pixels y-coordinate to an absolute measurement e.g. cm or inch. Because

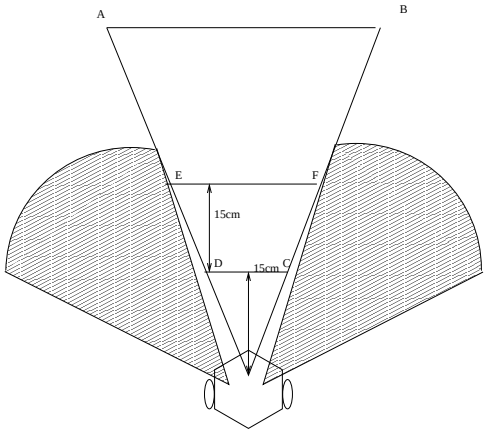


Figure 2: A visualisation of the monitored area. ABCD : the area captured by the camera. Shaded areas represent the sonar sensors views. Segment EF is the trigger distance line

the resolution of the images is very low, the distance estimation is not very accurate. At the lowest row of the image, where the ratio between pixel and the projected real world area is highest, each pixel represents an area of $2 * 1.5cm^2$.

The distance that triggers the robot to turn is set to 30 *cm*. The robot needs to turn fast enough so that the object will not be closer than 15 *cm* in front of it since the distance of any object in this area can not be calculated correctly. At maximum speed , the robot will have about two seconds to react and if the robot has already slowed down while approaching the object, it will have about three seconds. We have tried many different combinations of trigger distances and turning speeds to achieve a desirable combination. The first criteria is that the robot must travel safely, this criteria sets the minimum turning speed and distance. The width of view of the camera at the distance of 30 *cm* from the robot or 35 *cm* from the camera is 30 *cm*. The width of our robot is 20 *cm*, so if the vision module does not find an obstacle inside the trigger range, the robot can safely move forward. The second criteria is the robot needs to be able to go to cluttered areas. This means it should not turn too early when approaching objects. Also when the robot is confronted by the wall or a large object, it should turn just enough to move along the wall/object and not bounce back. This criteria encourages the robot to explore the environment.

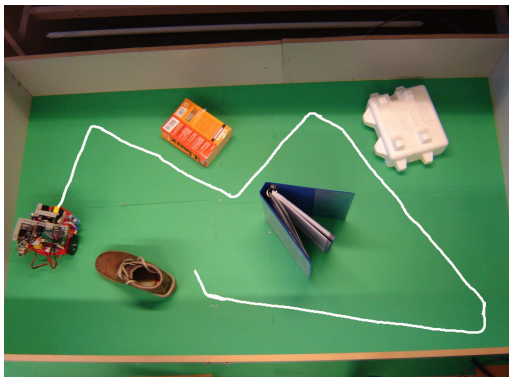
The main drawback of the no map approach is that the robot might get stuck in a confined or looped area. A common solution and opposite to ours is to build a model of the environment surrounding the robot. The model is a global map and can be used for planning a desirable path. Also the model can give the robot a virtual wider angle field of view by remembering objects that a robot saw from the previous experience. However with our robot configuration, the reactive mechanism can solve this problem. Since the robot can turn on the spot, we can guarantee that the robot will not be trapped indefinitely.

4 Experiments

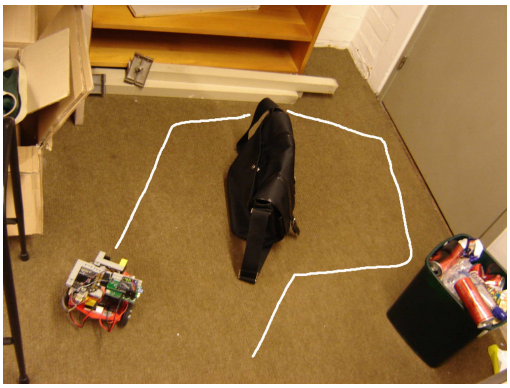
4.1 Experiment Setup and Results

We tested the robot in two environments, a $1.5 * 2.5m^2$ artificial arena surrounded by $30cm$ height walls and an office at the University of Kent Computing department, shown in Fig. 3. The surface of the artificial arena is a flat cartoon board with green wallpapers on top. We put different real objects such as boxes, shoes, books onto the arena. We first tested the robot in the arena with no objects (the only obstacless are walls) and then made the tests more difficult by adding objects. The office is covered with a carpet. The arena presents a more controlable environment where the surface is smooth and relatively colour-uniformed. The office environmnent is more challenging where even though the ground is flat its surface is much more coarse and not colour-uniformed.

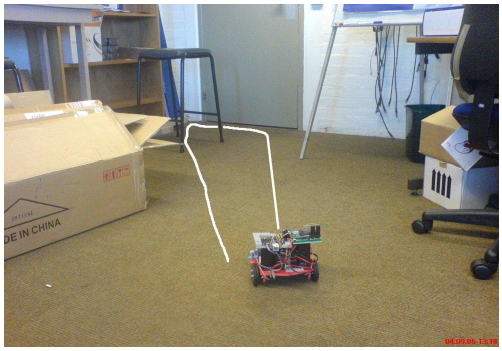
For each test, the robot run for 5 mins. We placed the robot in different places and put different objects into the test area. In general, the robot is quite competent; Table I summaries the result. The vision-based obstacle detection module correctly identified obstacle with almost 100% accuracy, that is if there was an obstacle in the camera view, the algorithm would register an non-ground area. Although the calculated distances of obstacles are not very accurate, they provide enough information for the controller to react. The simple mechanism of finding an open space worked surprisingly well. The robot was good at finding a way out in a small area such as the areas under tables and between chairs. The number of false positives are also low and only occured in the office environment. This is because the office's floor colours are more difficult to capture thoroughly. Further analysis revealed that false positives often occurred in the top part of the images. This is explained by the ratio of pixels/area in the upper part of the image being lower than the bottom part. At the top row of the image, each pixel corresponds to an area of $7 * 4cm$ while at the bottom row the area is $2 * 1.5cm$. Fortunately, the upper part also corresponds to the further area in real world. Therefore, most false positive cases resulted in unnecessary decreasing of speed but not changing direction. Because of the robot's reactive behaviour, it is capa-



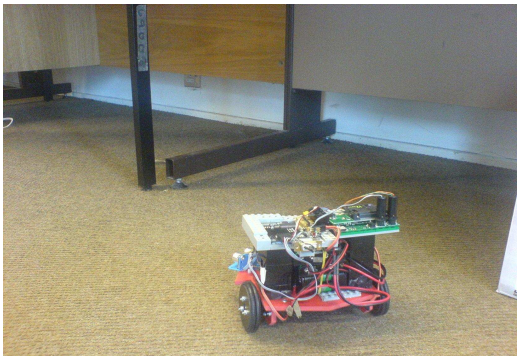
A



B



C



D

Figure 3: Snapshots of the robot in the test environments and its trajectories. A: the artificial arena with 4 objects. B: A small area near the office corner. C: A path that went through a chair's legs. D: An object with no base on the ground.

Table 1: Performance summary

Environment	No of Obstacles	Duration	Average speed	No. of collisions	False positive
Arena	0	60 min	13 cm/c	0	0%
Arena	1	60 min	10 cm/s	0	0%
Arena	4	60 min	6 cm/s	2	0%
Office	> 10	180 min	9 cm/s	7	3%

ble of responding quickly to changes in the environments. During some of the tests, we removed and put obstacles in front of the robot. The robot can instantly recognise the changes and changed it's course accordingly.

There are a small number of cases where the robot collided. Most of the collisions occurred when there are close objects in the direction that the robot was turning into and the sonar sensor failed to report the obstacles. Also there are objects that do not have their bases on the ground for the robot to see, in those situations there were miscalculations of the distances to the objects and hence collisions. In our environment, one of the objects that presented this problem was table cross-bar, (Fig. 3 D)

Fig. 3 shows 4 snapshots of the robot during operation and its trajectory. In picture A, the robot ran in the arena with 4 obstacles, it successfully avoided all the objects. On picture B, the robot went into a small area near a corner with a couple of obstacles and found a way out. On picture C, the robot successfully navigated through a chair's legs which presented a difficult situation. Picture D was a case where the robot failed to avoid an obstacle. Because the table leg cross-bar is off the floor, the robot underestimated the distance to the bar.

4.2 Discussion

We found that adding more obstacles onto the arena did not make the number of collisions increase significantly. However the average speed of the robot dropped as the arena get more crowded. The speed loss is admitedly due to the robot's reactive behaviour. The robot does not have a path planning therefore it can not always select the best path and only reacts to the current situation. In some cluttered areas, the robot spent a lot of time spinning around before it can find a viable path. We can improve the robot's behaviour in these situations by having a mechanism to detect cluttered and closed areas so the robot can avoid them. The assumption that the travelling surface is flat holds for most indoor environments. However, there a few objects that do not have their bases on the ground or protrude from their bases. Another sonar sensor pointing at the front of the robot will solve this problem. The additional sensor can also be used for auto-calibarating the vision module.

Each control cycle takes about 150*ms* or 7Hz. Table II shows the time spent on each task in the cycle. Most of the times is spent waiting for the camera images and sonar data. The algorithm used only 15% of the CPU during operation. This leaves plenty of resources for higher behaviours if needed. It is possible to implement this algorithm with a less powerful CPU. Since only 10% of the CPU time is spent on processing data, a CPU running at 20 MHz would be sufficient. So instead of the Gumstix computer we can use a micro-controller such as a Brainstem or a BASIC STAMP for both image processing and motion control without any decrease in performance. The memory usage is nearly one Mb which is rather big. We did not try to optimise memory usage while implementing the code so improvements could be made. We plan to implement this control algorithm on a micro-controller instead of the Gumstix. This change will reduce the cost and power usage of the robot by a large amount. To the best of our knowledge, there has not been a mobile robot that can perform reliable obstacle avoidance in unconstrained environments using such low resolution vision and slow microprocessor. A robot with only obstacle avoidance behaviour might not be very useful apart from applications such as exploration or surveillance. However given that our algorithm is very computationally efficient and requires low resolution images, it can be incorporate into a more complex system at a small price and leaves plenty of resources for higher level behaviours.

Table 2: Speed performance

Task	Time
Image acquiring	95 ms
Sonar sensor reading	125 ms
Image processing	5 ms
Controller	1 ms
Servos updating	$< 1ms$
Logging	3 ms
Total	150 ms

5 Conclusion and future research

We have developed a vision-based obstacle avoidance algorithm that is efficient enough to run on a small computing platform. Our robot uses digital camera as the main sensor which is usually only available for bigger robot platforms. Even when the visual image resolution is decreased dramatically, by factor of 16, it still provides enough information for detecting obstacles and a robust behaviour. The camera shortcoming of narrow FOV is compensated by sonar sensors. Our robot control strategy consists of several reactive rules. No world model is built, in-

stead the controller only reacts to immediate sensor measurements. The obstacle avoidance strategy is derived from the combination of the robot's dynamics and sensor setting. The robot was tested in a real office environment and performed very well.

One of the biggest disadvantages of colour-based object detection is the need of calibrating the vision module before operation. It is very desirable to have a robot that can be deployed in any environment that meets some specific requirements without preparation. We are implementing a simple auto-calibrating procedure on-board the robot. This procedure will be run before each operation. The robot must be deployed in a place where the immediate area in front of it is free of obstacles. The robot will move forward for a few seconds and take pictures of the floor for the calibration process. Another approach is to add another sonar sensor pointing forward with the camera. The correlation between this sonar measurement and the visual module output can be learned. This correlation then can be used for auto-calibrating the visual module.

REFERENCES

http://www.gumstix.org.

Batalin, M., Shukhatme, G., and Hattig, M. (2003). Mobile robot navigation using a sensor network. *IEEE International Conference on Robotics & Automation (ICRA), 2003*.

Braitenberg, V. (1984). *Vehicles: Experiments in Synthetic Psychology*. MIT Press/Bradford books.

Brooks, R. A. (1985). A robust layered control system for a mobile robot. Technical report, MIT, Cambridge, MA, USA.

Ibanez-Guzman, J., Jian, X., Malcolm, A., Gong, Z., Chan, C. W., and Tay, A. (2004). Autonomous armoured logistics carrier for natural environments. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 473–478.

Lenser, S. and Veloso, M. (2003). Visual sonar: fast obstacle avoidance using monocular vision. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 886–891.

Lorigo, L., Brooks, R., and Grimsou, W. (1997). Visually-guided obstacle avoidance in unstructured environments. In *Intelligent Robots and Systems, 1997. IROS '97., Proceedings of the 1997 IEEE/RSJ International Conference on*, volume 1, pages 373–379, Grenoble, France.

Lu, Y., Zhang, J., Wu, Q., and Li, Z.-N. (2004). A survey of motion-parallax-based 3-d reconstruction algorithms. *Systems, Man and Cybernetics, Part C, IEEE Transactions on*, 34(4):532–548.

Manduchi, R., Castano, A., Talukder, A., and Matthies, L. (2005). Obstacle detection and terrain classification for autonomous off-road navigation. *Autonomous Robot*, 18:81–102.

Santos-Victor, J., Sandini, G., Curotto, F., and Garibaldi, S. (1993). Divergent stereo for robot navigation: learning from bees. In *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR '93., 1993 IEEE Computer Society Conference on*, pages 434–439, New York, NY.

Stentz, A., Kelly, A., Rander, P., Herman, H., Amidi, O., Mandelbaum, R., Salgian, G., and Pedersen, J. (2003). Real-time, multi-perspective perception for unmanned ground vehicles. *AUVSI*.

Thrun, S., Bennewitz, M., Burgard, W., Cremers, A., Dellaert, F., Fox, D., Hahnel, D., Rosenberg, C., Roy, N., Schulte, J., and Schulz, D. (1999). MINERVA: a second-generation museum tour-guide robot. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 3, pages 1999–2005, Detroit, MI, USA.

Ulrich, I. and Nourbakhsh, I. R. (2000). Appearance-based obstacle detection with monocular color vision. In *AAAI/IAAI*, pages 866–871.

Zufferey, J. and Floreano, D. (2005). Toward 30-gram Autonomous Indoor Aircraft: Vision-based Obstacle Avoidance and Altitude Control. In *IEEE International Conference on Robotics and Automation (ICRA'2005)*. Sponsor: Swiss National Science Foundation.