Detecting Motion in the World with a Moving Quadruped Robot

Peggy Fidelman[1], Thayne Coffman[2], Risto Miikkulainen[1] and Peter Stone[1]

[1] Department of Computer Sciences
The University of Texas at Austin
[2] Department of Electrical and Computer Engineering
The University of Texas at Austin

Abstract. For a robot in a dynamic environment, the ability to detect motion is crucial. Although legs are arguably the most versatile means of locomotion for a robot, and thus the best suited to an unknown or changing domain, existing methods for motion detection either require that the robot have wheels or that its walking be extremely slow and tightly constrained. This paper presents a method for detecting motion from a quadruped robot walking at its top speed. The method is based on a neural network that learns to predict optic flow at each timestep, thus allowing anomalies to be detected. The system is demonstrated to be capable of detecting motion in the robot's surroundings.

1 Introduction

The ability to detect motion is important to a robot in a novel or changing environment. Motion can potentially be a very large clue about which parts of the environment are interesting or dangerous. For example, consider a consumer robot in the home, such as the commercially available Sony Aibo[1]. Motion can give it clues to where humans are located in its environment, which will help it interact with them more effectively. It can also be used to direct the robot's attention to potential danger, such as a stack of books sliding off of a desk or the family dog preparing to pounce on it. As another example, consider a surveillance robot [2]. The importance of motion detection in this case should be readily apparent, since this is the primary mechanism on which modern surveillance systems are based.

If a robot is to be able to deal with truly novel environments, it is also an advantage for the robot to have legs rather than wheels. Legs allow traversal of highly uneven surfaces. A legged robot can step over obstacles or climb stairs, whereas analogous feats are often impossible for a wheeled robot. However, the motion of a legged robot is not as smooth as that of a wheeled robot. This introduces additional challenges for detecting motion in the world from such a robot while it is moving.

The main contribution of this paper is a method for detecting motion from a quadruped robot while it is walking at its maximum speed (approximately 34cm/sec). Our method is inspired by that of Lewis [3], but differs in several

ways and is shown to be effective in a less constrained environment with a significantly faster-moving robot.

The remainder of this paper is organized as follows. Section 2 gives a background and describes related work. Section 3 details our method for detecting external motion. Section 4 describes the setup of the system used in the experiments, as well as the details of the experiments themselves. Section 5 shows the results of these experiments, and Section 6 discusses these results as well as possible directions for the future. Section 7 concludes with a brief examination of the contributions of this work.

2 Background and Related Work

2.1 Optic Flow

Optic flow is a way of describing the apparent motion between two images of the same scene taken in quick succession. It is typically expressed as a vector field, with a two-dimensional vector for each pixel in the first image, representing vertical and horizontal displacement. These vectors give a complete description of where each pixel in the first image appears to have moved to in the second image.

The relationship between the apparent motion in two dimensions and the actual motion in three dimensions is non-trivial. Sometimes it is difficult to determine whether the motion is the result of the camera moving or objects in the scene moving. In other cases, it is clear that an object is moving and not the camera, but the actual direction of the object's motion cannot be determined because only part of the object is visible (this is known as the aperture problem).

For an intuitive understanding of some of these issues, consider a passenger looking out the right-hand side window of a car at an adjacent vehicle while s/he is waiting for a stoplight to turn green. Without prior knowledge, the scenes s/he perceives if the car appears to move to the left could be interpreted in a number of ways – the other car might be moving forward and the passenger's car might still be stationary, or the other car might be stopped and our passenger might be moving backwards, or both cars might be moving (forward or backwards) at different rates. It is clear that while the optic flow field contains a wealth of knowledge, interpretation of those fields can be difficult.

Optic flow is formulated in terms of instantaneous (in space and time) image intensity gradients. The key formula defining the optic flow between two images in a sequence is

$$I_{t+dt}(x + u(x, y), y + v(x, y)) = I_t(x, y), \tag{1}$$

where $I_t(x, y)$ is the image intensity at each pixel at time t, $I_{t+dt}(x, y)$ is the image intensity at time $(t + dt)$, $u(x, y)$ is the horizontal flow at each pixel, and $v(x, y)$ is the vertical flow at each pixel. Thus the horizontal and vertical optic flow generate a "mapping" between corresponding pixels in the two images.

A perfect solution to the optic flow formula is rarely available because of noise in the input images and other effects like occlusion. Instead of looking for a total solution, most approaches attempt to minimize the error in equation (1) summed over all pixels in the image. Computation of optic flow is an under-constrained problem, so in addition to minimizing the error in the pixel matching between images, a smoothness constraint is typically added. This constraint is justified because for images of real-world objects (which are, in general, smooth and connected) we can expect the optic flow field to be smooth at almost every pixel. The objective function for optimization then becomes

$$\sum_{(x,y)} E^2(x,y) = \sum_{(x,y)} \left\{ (I_x u + I_y v + I_t)^2 + \lambda(u_x^2 + u_y^2 + v_x^2 + v_y^2) \right\}$$

with

$$f_x = \frac{\partial f}{\partial x} \quad , \quad f_y = \frac{\partial f}{\partial y} \quad , \quad f_t = \frac{\partial f}{\partial t}.$$

The correctness and smoothness components to the optimization are clearly visible as the first and second group of terms in the objective function, respectively. As the optic flow field generates a more accurate matching between pixel intensities in the two images, the first group of terms will decrease towards zero. As the field becomes smoother (showing less variation between adjacent pixels), the second group of terms will decrease towards zero. The relative importance of these terms is regulated by the parameter λ. Because its formulation is based on local gradient information, computation of the optic flow field is often more accurate when images are only incrementally different.

Optic flow is often computed via iterative relaxation, by one variation or another of an approach developed by Horn and Schunk in the 1980's [4]. In this approach, the proposed solution is initialized, and on each iteration, the solution is refined by propagating information from each pixel to its local neighbors through a local averaging of the optic flow field. This process is guided by the equations

$$u^k(i,j) = \bar{u}^{k-1}(i,j) - I_x(i,j)\frac{P(i,j)}{D(i,j)}$$

$$v^k(i,j) = \bar{v}^{k-1}(i,j) - I_y(i,j)\frac{P(i,j)}{D(i,j)}$$

$$P = I_x \bar{u} + I_y \bar{v}$$

$$D = \lambda^2 + I_x^2 + I_y^2$$

In these equations, $u^k(i,j)$ and $v^k(i,j)$ are the kth iteration's estimates of the horizontal and vertical flow at pixel (i,j), and $\bar{f}(x,y)$ is the local average of function $f(\cdot)$ near pixel (x,y). Iteration continues until a convergence condition is reached or a maximum number of iterations, $k_{\max}$, have occurred. This 25-year-old approach continues to be one of the more common and effective methods of optic flow computation, although many alternatives have been studied.

2.2 Optic Flow in Navigation

Optic flow can be extremely useful in navigation and obstacle detection, and several groups have used it for these purposes in robots [5, 6]. In general, the robots are wheeled (or airborne) instead of legged, which means that their typical motion is smoother. This makes normal optic flow easier to characterize, so abnormalities such as the ones caused by obstacles can readily be detected.

A method for detecting obstacles in the walking path of a bipedal robot using optic flow information was developed by Lewis [3]. In this method, a neural network uses the robot's joint angles and gait phase to predict optic flow events. This prediction is compared to observed optic flow events, and any significant difference between the two indicates that an obstacle has been detected.

The success of Lewis's approach indicates that optic flow can be a source of useful information even on legged robots. It also indicates that neural networks have promise as a tool for overcoming the difficulty of characterizing normal optic flow on legged robots, as discussed above. But Lewis's experiments take place in a very constrained environment. The robot is tethered so that it walks in a circle, and its camera is fixed at a slight downward angle so that it is always focusing on the ground slightly in front of it, which is the same at all points around the circle (with the exception of the obstacles it must detect during the testing phase). It also walks very slowly (2cm/sec).

In order to be generally useful, a motion detection method for a legged robot must not be subject to these constraints. If the robot cannot move freely through non-uniform environments and still detect motion, there is little benefit to using a legged robot at all – a wheeled robot or even a stationary camera could probably do the same task more reliably. In addition, a robot should ideally not have to slow down its own movements in order to accommodate the motion detection algorithm. The method presented here is effective on a freely moving robot walking at its top speed (34cm/sec). The chief ways in which it differs from Lewis's approach are: i) it operates on the raw optic flow field rather than on preprocessed data, or optic flow "events"; and ii) a substantial postprocessing step has been added as part of comparing the neural network's prediction to the actual observed optic flow. It is not clear to what extent our predictor neural network differs from that of Lewis, because details of his architecture are not available.

3 Method Architecture

Our method is depicted in Figure 1. First, the optic flow in the image is calculated. The resulting vector field is then given as input to a neural network along with information about the current position in the robot's walk cycle and readings from its three accelerometers. This neural network outputs a prediction of the optic flow to be seen in the next image. When the robot receives this next image, the optic flow is calculated and compared with the network's prediction. A postprocessing algorithm then looks for discontinuities in the difference be-

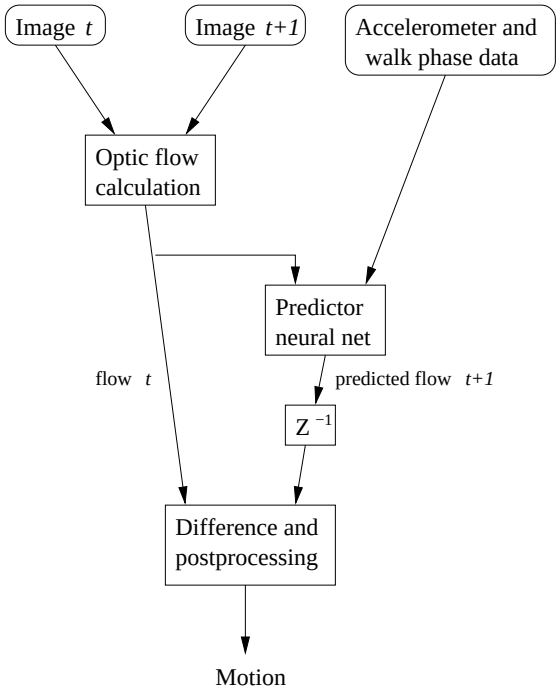

Fig. 1. The architecture of our method for detecting moving objects in the robot's environment. Z^{-1} is a one-step delay operator.

tween the calculated and predicted optic flow to determine where in the image externally moving objects are present.

3.1 Optic Flow

We did not compute optic flow in real-time on the Aibo. Instead, training and testing sequences alike were transferred to a set of image files on a desktop PC's hard disk. Image sequences were then loaded off the disk, the optic flow calculations were run, and the resulting flow fields were stored back to the disk for later use in training or comparison against predicted flow.

The optic flow fields for training the neural network and for comparing against predicted output were computed by a Matlab implementation of Horn & Schunk's iterative relaxation algorithm described above. Identical parameters, $\lambda = 1.0$ and $k_{\max} = 1000$, were used to compute all training and testing data. Our choice of optic flow parameters resulted in very smooth fields. While generally smooth fields might be expected given the Aibo's known motion characteristics and environment, additional work will be performed in the future to understand whether using parameters that generate more rapidly-varying optic flow fields could allow more accurate motion discrimination.

3.2 Predictor Neural Network

The neural network shown in Figure 1 accepts three types of information as input. The first is an optic flow field. The second is a single number indicating the robot's position in the walk cycle at the time of the second of the two images used to calculate this optic flow field. The third is a set of accelerometer data corresponding to the same image. In our experiments, we use a simple recurrent network architecture [3] (see Figure 2).

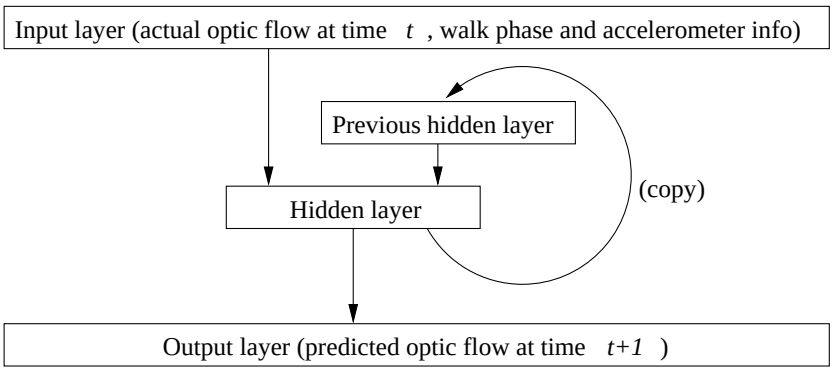

Fig. 2. The simple recurrent network used to predict optic flow. All projections are full and feed-forward, except for the projection from the hidden layer to the previous hidden layer. This arrow instead indicates a direct copy of the hidden layer contents into the previous hidden layer. In this way, some context is kept in the network.

The network is trained on sequences of optic flow fields, which are generated from sequences of robot camera images in which there is no external motion. Thus, the network is effectively learning to produce what the next optic flow field should look like *if there is no external motion*, given the last optic flow field observed and information about the robot's acceleration and position in the walk cycle. By comparing this prediction to the actual optic flow observed at the next time step during performance, we can see which parts of the image exhibit unpredictable motion, which gives us information about where objects moving relative to the world can be found in the image.

3.3 Postprocessing

Consider the vector field resulting from taking the (vector) difference of the predicted and actual optic flow fields. If we take the magnitude of each of the

[3] Informal experimentation indicated that this gives some improvement in performance over a three layer feed-forward network. However, such a non-recurrent network is capable of some success at this task.

elements of this vector field, we obtain a matrix of the same size, which we will refer to as the *difference field*.

The discontinuous motion of the robot's camera and the noisy nature of real-world sensor data make the optic flow prediction task quite difficult. Although the neural network typically does a reasonable job of predicting optic flow for image sequences containing no external motion, its prediction is occasionally entirely wrong. This means that the naive approach – simply taking the size of the difference field at each point to be the likelihood of external motion at that point – will not suffice.

However, these magnitudes do give us some useful information. Because the neural network is trained with images from a world in which nothing other than the robot itself is moving, all of its targets during training were optic flow fields with coherent motion. So, assuming the images contain no external motion, even when the network makes a wrong prediction that prediction will be more or less "equally wrong" at all points. If there is external motion in the scene, however, there will often be sharp discontinuities in the difference field, which can be discovered by running an edge detection algorithm on each difference field.

The edge detection we use in our postprocessing step is rather unconventional. Many edge detection algorithms are designed to find the best edges in an image, even if that image has only poor candidates for edges. However, if there are no sharp edges in the difference field at some timestep, this probably indicates that there is no motion in the image. Therefore, in this case we want our postprocessing algorithm not to find any edges. To this end, we first make a binary version of the difference field by finding its maximum element and then replacing every element which is extremely far away from this maximum[4] with a zero and replacing the rest with ones. Then a conventional edge detection algorithm is run over this binary difference field (in our experiments we use the Laplacian of Gaussian method).

4 Experimental Procedure

The images used in our experiments all come from a Sony Aibo ERS-7 walking across a standard 2004 RoboCup legged league field [7]. The Aibo has three degrees of freedom in each of its four legs as well as its head. It has a CMOS camera in the head, from which we are able to capture approximately 25 images per second. The robot always starts close to the center of the yellow goal facing outward toward the field center (although not always in the *exact* same location) and then walks most of the way across the field[5] using the fastest available forward walk (approximately 34cm/sec). Due to the Aibo's slight left-right weight

[4] In practice, to be "extremely far away from the maximum," an element must be very close to zero and the maximum over the difference field must be quite large. This enforces that all edges which will be found in the next step will correspond to very sharp edges in the original difference field.

[5] The length of these trajectories is constrained by the amount of memory available on the Aibo.

asymmetry, the forward walk curves slightly to the right over long distances. Thus a typical trajectory looks like the one depicted in Figure 3.

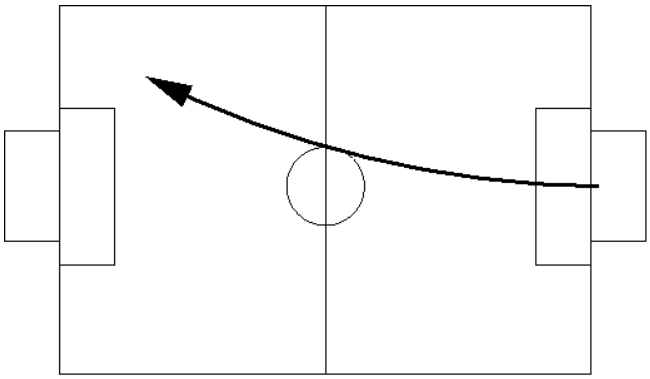

Fig. 3. A typical trajectory used in our experiments.

Instead of working with the full Aibo imaging resolution, we first reduced the resolution by a factor of 6 in both the horizontal and vertical directions. We did this by averaging 3-by-3-pixel blocks of half-resolution Aibo images. Images were converted from full color to grayscale images before the optic flow computation, but no other image preprocessing (histogram equalization, deblurring, etc.) was performed.

As a result of the resolution reduction, the images and flow fields were 35 columns by 27 rows. Our intent in reducing the image resolution was to reduce the complexity of the neural network that needed to be trained. As a side effect, however, the optic flow computations also ran faster. Exact runtime performance was not recorded, but computing the optic flow field for one frame pair required approximately 1s of CPU time on a 1.8GHz desktop machine. Real-time performance will require that our implementation be translated from Matlab to a language more suitable for embedded operation on the Aibo, and that much more attention be paid to efficient computation.

As discussed in Section 3.2, we used a simple recurrent architecture for the predictor neural network. In our experiments, this network had a 200-unit hidden layer. It was trained with backpropagation (using momentum) on data from six runs of the robot on an empty field, where each run consisted of approximately 150 sequential images. Training of this network took approximately 1050 epochs.

5 Results

Figure 4 shows four sequential frames from one testing run of the system. The robot in the foreground is moving at full speed; all other parts of the image are stationary relative to the world. For comparison, Figure 5 shows some typical

Fig. 4. Four sequential frames from a testing run. The robot in the images is moving to the left at full speed.

Fig. 5. Some typical errors on a field with no motion. In the run from which these images were taken, in which there was no motion on the field, 83% of the frames were correctly labeled as containing no motion.

errors from a testing run with no external motion. In this testing run, 83% of the frames were correctly labeled as containing no motion.

To judge the classification accuracy of our system, we applied our system to four sets of images from trajectories of the sort shown in Figure 3. Two of these runs contained one moving robot, one contained three moving robots, and one contained no external motion. Each image in the four sets was divided into 9 sectors (see Figure 6), and each sector was labeled by hand as containing motion or not. If less than 1/4 of the sector contained a moving object, the sector was labeled as not containing motion. This ground truth was compared to the motion detected by the system. Results are shown in Figure 7 and Figure 8. Note that the postprocessing parameters have been tuned to an extreme which eliminates as many false positive motion detections as possible. It is possible to improve the classification accuracy of sectors containing motion by setting these parameters to a less extreme value; however, this reduces the classification accuracy of sectors not containing motion.

6 Discussion and Future Work

Observation has shown that the system detects motion more reliably when the moving object is closer to the robot and moving more rapidly. Although this has not been verified quantitatively, it would be a good topic for future investigation. The fact that it is hard for the system to detect objects that are moving slowly is understandable in light of the fact that the robot's own motion is so rapid. Additionally, the fact that proximity helps motion detection is reasonable not only because near motion will appear more rapid in two dimensions, but also

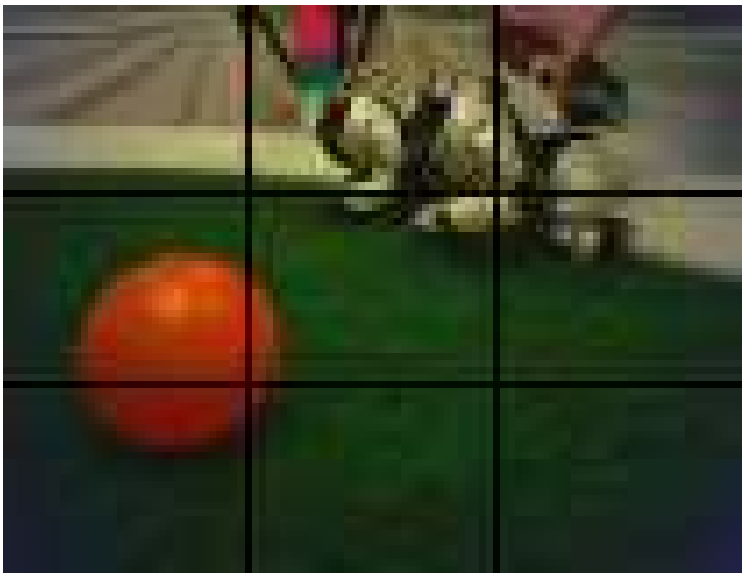

Fig. 6. An example image showing division into sectors. The middle sector would not be labeled as containing motion, because less than 1/4 of it contains moving objects.

because our system downsamples images so much – a moving object sufficiently far away will appear as a single pixel whose color is changing slightly.

This suggests that a fruitful direction for the future may be to expand the system to work with larger images. Although this will greatly increase the size of the predictor neural network, we believe that this will not make training intractable, in light of the fact that training time for the network at its current size only requires a few hours of CPU time. We have already discovered that optic flow calculation over a half-resolution image (as opposed to the 1/6 resolution images we are currently using) is reasonable.

Another direction for future work is real-time implementation on an Aibo robot. This is actually quite plausible, given the size of the images we are working with. The Aibo has a 576 MHz 64 bit RISC processor which allows for significant amounts of onboard computation. Although the computation required to process the images would reduce the rate at which we can capture images, it is likely that a slower gait could compensate for this.

7 Conclusion

A method has been presented for detecting external motion from a quadruped robot while it is walking freely and quickly. It has been demonstrated that this method can successfully detect such motion. Although the system does not run in real-time onboard the robot, it is able to handle real-world sensor data, and

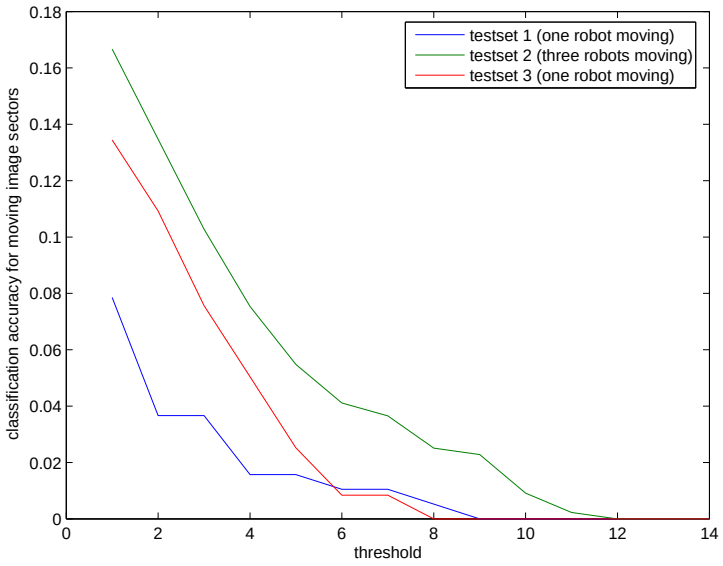

Fig. 7. Classification accuracy for image sectors containing motion. Parameters have been tuned to decrease false positives as far as possible. The x-axis corresponds to the number of edge pixels in a sector required for motion to be detected in that sector.

it is able to process this data with a speed which indicates that future onboard implementation of our method is not outside the realm of possibility.

Acknowledgments

Thanks to Mohan Sridharan for his help implementing edge detection and to Uli Grasemann for his help with image dimensions. Thanks also to the UT Austin Villa robot soccer team for their efforts in creating the Aibo code base which supports this project.

References

1. Sony: Aibo robot (2004) `http://www.sony.net/Products/aibo`.
2. Rybski, P.E., Stoeter, S.A., Erickson, M.D., Gini, M., Hougen, D.F., Papanikolopoulos, N.: A team of robotic agents for surveillance. In: Proceedings of the Fourth International Conference on Autonomous Agents. (2000)
3. Lewis, M.A.: Detecting surface features during locomotion using optic flow. In: Proceedings of the IEEE International Conference on Robotics and Automation, Washington, DC (2002)

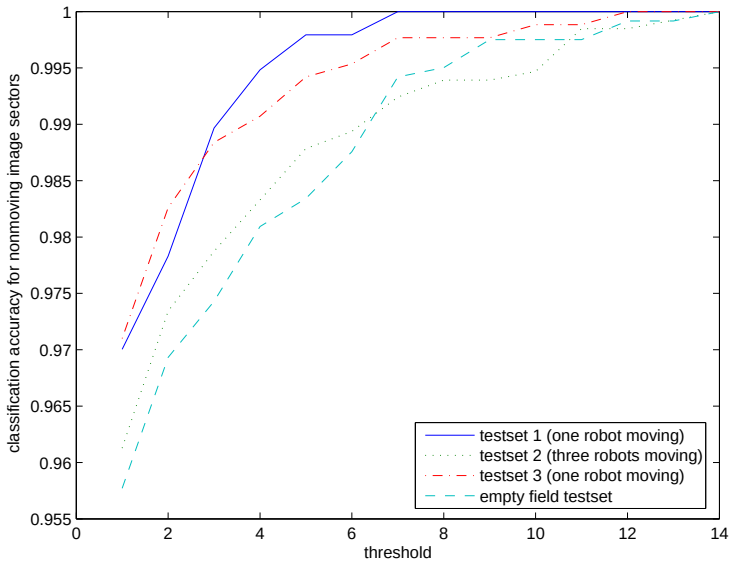

Fig. 8. Classification accuracy for image sectors not containing motion. Parameters have been tuned to decrease false positives as far as possible. The x-axis corresponds to the number of edge pixels in a sector required for motion to be detected in that sector.

4. Horn, B.K.P., Schunck, B.: Determining optical flow. Artificial Intelligence **17** (1983) 185–203
5. Bhanu, B., Das, S., Roberts, B., Duncan, D.: A system for obstacle detection during rotorcraft low altitude flight. IEEE Transactions on Aerospace and Electronic Systems **32** (1996) 875–897
6. Young, G.S., Hong, T.H., Herman, M., Yang, J.C.: Safe navigation for autonomous vehicles: a purposive and direct solution. Proceedings of SPIE: Intelligent Robots and Computer Vision XII: Active Vision and 3D Methods **2056** (1993) 31–42
7. RoboCup Technical Committee: Sony four legged robot football league rule book (2004) `http://www.tzi.de/~roefer/Rules2004/Rules2004.pdf`.