Detecting Motion in the Environment with a Moving Quadruped Robot

Peggy Fidelman[1], Thayne Coffman[2], and Risto Miikkulainen[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. Motion often indicates areas of the robot's surroundings that are changing, contain another agent, or are otherwise worthy of attention. 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 caused by its walk, thus allowing environment motion to be detected as anomalies in the flow. The system is demonstrated to be capable of detecting motion in the robot's surroundings, forming a foundation for intelligently directed behavior in complex, changing environments.

Keywords: robot vision, image processing.

1 Introduction

The ability to detect motion is important to a robot in a novel or changing environment. Motion can potentially be a very significant 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]. Modern surveillance systems are primarily based on motion detection. If a robot must stop its own motion in order to detect motion in its surroundings, much of the advantage of using a robot instead of a simpler system (such as a set of stationary cameras) is lost.

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 characteristic introduces additional challenges for detecting motion in the environment while the robot itself 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 35cm/sec). The overall architecture is inspired by that of Lewis [3], but our method differs in several important ways: preprocessing of optic flow has been eliminated, a substantial postprocessing step has been added to the output of the neural network, and the details of most of the architecture's components have been redesigned. These innovations make our method effective in a less constrained environment than Lewis's method requires, and also allow use of a significantly faster-moving robot.

The paper is organized as follows. Section 2 gives a background on optic flow and describes related work that uses optic flow for navigation and obstacle detection on mobile robots. Section 3 introduces the 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 presents the results of these experiments, and Section 6 discusses these results as well as possible directions for the future.

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 in the second image.

The apparent motion in two dimensions depends on the actual motion in three dimensions in a complex manner. 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 effect 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 the passenger might be moving backward, or both cars might be moving forward or backward at different rates. Thus, while the optic flow field contains a wealth of knowledge, interpretation 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 flows generate a "mapping" between corresponding pixels in the two images.

A perfect solution to the optic flow formula is rarely available because of effects such as noise and occlusion. Instead of trying to compute 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 underconstrained problem; therefore, in addition to minimizing the error in the pixel matching between images, a smoothness constraint is typically included. This constraint is justified because for images of real-world objects (which are, in general, smooth and connected) the optic flow field is likely 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

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

where $f \in \{I, u, v\}$. 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 formulation of the optic flow field is based on local gradient information, its computation 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 1980s [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. Although many alternatives have been studied, this 25-year-old approach continues to be one of the most common and effective methods of optic flow computation.

2.2 Optic Flow in Navigation

Optic flow is 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 regularity of motion 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 recently 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 suggests that neural networks are a promising tool for overcoming the difficulty of characterizing normal optic flow on legged robots, as discussed above. However, Lewis's experiments take place in a highly 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 should be free of 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 in this paper is effective on a freely moving robot walking at its top speed (35cm/sec). It also differs from Lewis's approach in that it operates on the raw optic flow field rather than on preprocessed data (optic flow "events"), and 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 the predictor neural network differs from that of Lewis, because details of his architecture are not available.

3 Motion Detection Method

The proposed 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 between the calculated and predicted optic flow to determine where in the image the externally moving objects are likely to be found.

3.1 Optic Flow

Training and testing sequences of optic flow were obtained from the Aibo and transferred to a set of image files on a desktop PC's hard disk. Image sequences

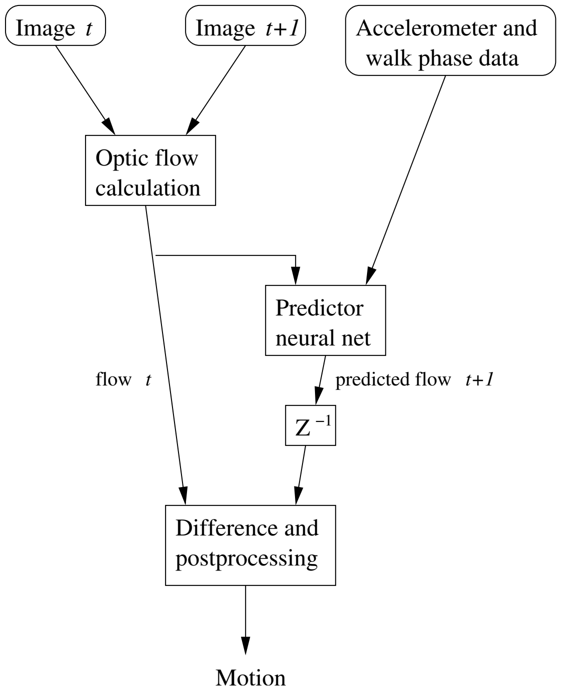

Fig. 1. An overview of the method for detecting moving objects in the robot's environment. (Z^{-1} is a one-step delay operator.) Optic flow, accelerometer readings, and information about the current phase of the robot's walk are given as inputs to a neural network, which then predicts the optic flow to be observed at the next timestep. The difference between the prediction and the observed optic flow is then used to detect any locations in the image that contain moving objects.

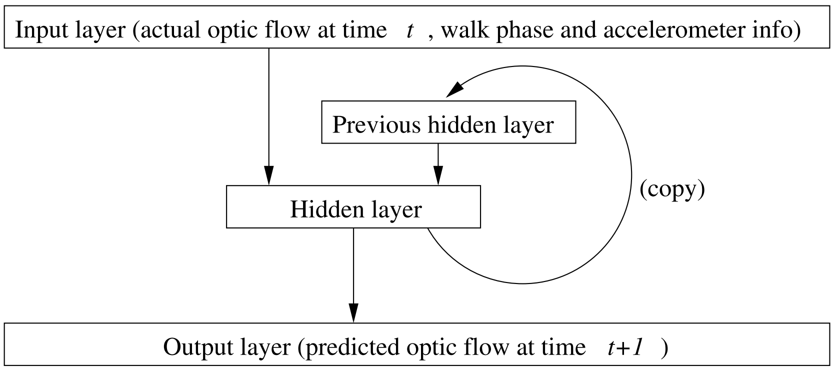
Input layer (actual optic flow at time t, walk phase and accelerometer info)
Previous hidden layer
(copy)
Hidden layer
Output layer (predicted optic flow at time t+1)

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

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. Although in principle these calculations could be done in real time on the Aibo, offline experiments test the method more efficiently.

The optic flow fields were computed by a Matlab implementation of Horn & Schunk's iterative relaxation algorithm described in Section 2.1. Identical parameters, $\lambda = 1.0$ and $k_{\max} = 1000$, were used to compute all training and testing data. These parameter choices resulted in very smooth fields. While generally smooth fields might be expected given the Aibo's known motion characteristics and environment, additional work will have to 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 predictor 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 image (of the two images used to calculate the optic flow field). The third is a set of accelerometer data corresponding to the same image. In the experiments described in this paper, a simple recurrent network architecture [7] was used [1] (Figure 2).

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

[1] Informal experimentation indicated that a simple recurrent network performs better than a three layer feed-forward network. However, such a non-recurrent network is also capable of some success at this task.

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 timestep, it is possible to see which parts of the image exhibit unpredictable motion, indicating 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. Taking the magnitude of each of the elements of this vector field results in a matrix of the same size, called 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. Therefore, 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 contain useful information. Because the neural network is trained with images from an environment 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 used in the 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, there is probably no motion in the image. Therefore, in this case the postprocessing algorithm should not find any edges. To this end, first a binary version of the difference field is obtained by finding its maximum element and replacing every element extremely far away from this maximum[2] with a zero, and replacing the rest with ones. Then a conventional edge detection algorithm (such as the Laplacian of Gaussian method) is run over this binary difference field.

4 Experimental Procedure

The images used in the experiments discussed here were acquired by a Sony Aibo ERS-7 walking across a standard 2004 RoboCup legged league field [8]. The Aibo

[2] 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 large. This constraint enforces that all edges found in the next step will correspond to very sharp edges in the original difference field.

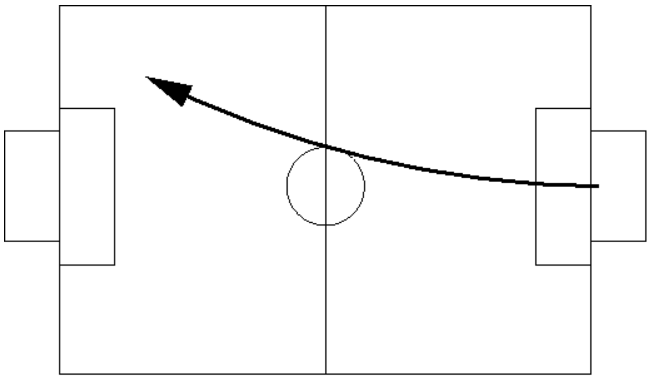

Fig. 3. A typical trajectory used in our experiments

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 it is possible to capture approximately 25 images per second[3]. 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[4] using the fastest available forward walk (approximately 35cm/sec)[9]. Due to the Aibo's slight left-right weight asymmetry, the forward walk curves slightly to the right over long distances. Thus a typical trajectory looks like the one depicted in Figure 3.

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

The resulting images and flow fields consisted of 35 columns by 27 rows. This lowered image resolution allowed a simpler neural network to be trained and decreased the runtime of optic flow computations. 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 the current implementation be optimized for speed and translated from Matlab to a language more suitable for embedded operation on the Aibo.

As discussed in Section 3.2, a simple recurrent architecture was used for the predictor neural network. 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.

[3] The hardware is capable of capturing 30 frames per second, but software overhead reduces this number somewhat.

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

5 Results

As was discussed in Section 1, motion detection on a robot is most useful for alerting the robot to potential anomalies in its environment, particularly those that suggest there are changes taking place. Thus, robots will typically want to react immediately as soon as any motion is detected; the value of such information decays very rapidly. However, this means false positive motion detections are particularly dangerous – each one is likely to distract the robot from its primary task unnecessarily. Thus, before incorporating this technology into any robotic system, it is important not only to demonstrate that it can provide useful information about motion, but also that it can do so while keeping false positives to a minimum.

The results shown here reflect these priorities. The parameters of the postprocessing algorithm were set to make the system maximally resistant to false positives. Then, to judge the classification accuracy of the system, it was applied 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 (Figure 4), 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.

The system correctly labeled a significant portion of the image sectors containing motion, despite the postprocessing parameter values that virtually eliminated

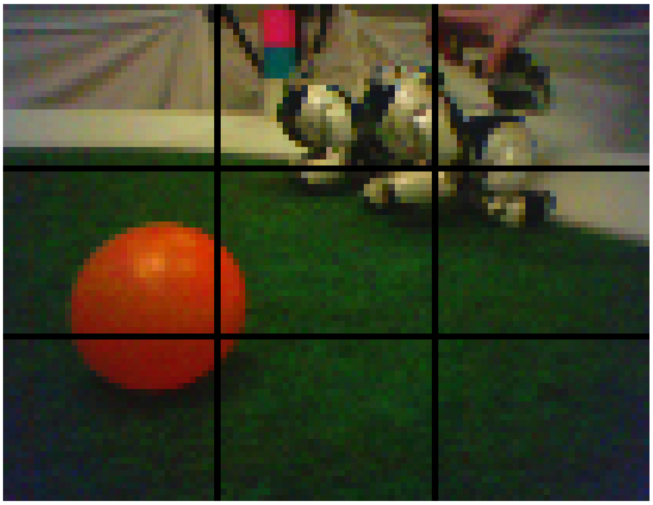

Fig. 4. An example image showing division into sectors, for use in the quantitative evaluation of the motion detection method. Though both robots and the ball in this image are moving, the middle sector would not be labeled as containing motion, because less than 1/4 of it contains moving objects.

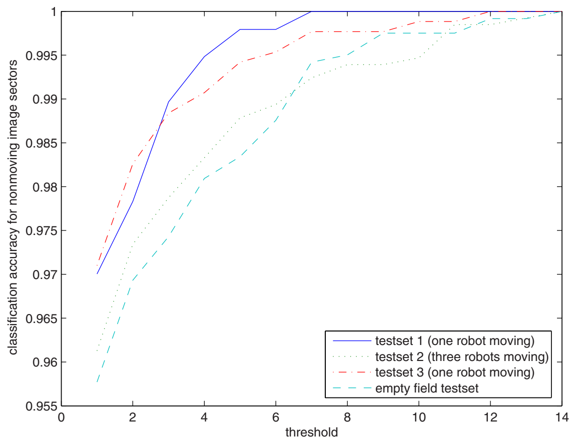

Fig. 5. Classification accuracy for image sectors not containing motion. The x-axis corresponds to the number of edge pixels in a sector required for motion to be detected in that sector, and the y-axis is the fraction of total image sectors not containing motion that were correctly labeled as not containing motion. As more edge pixels are required for a sector to be labeled as containing motion, accuracy in labeling non-moving sectors increases (i.e., false positives decrease), though there is a tradeoff with accuracy in labeling moving sectors (see Figure 6). Note, however, that even when this threshold is set to 1, accuracy is greater than 95% for all testsets.

false positive motion detections (see Figures 5 and 6). Note that if these parameters are set to less extreme values, classification accuracy of sectors containing motion can be improved, so in this sense Figure 6 reflects the "worst-case" result for motion detection. However, the current settings have the considerable advantage that image sectors labeled as containing motion are virtually certain to actually contain motion.

The typical qualitative behavior of the system is shown in Figures 7 and 8. Figure 7 contains four sequential frames from one test 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 8 shows typical errors from a testing run with no external motion. In this testing run, over 95% of the sectors were correctly labeled as containing no motion (see Figure 5).

6 Discussion and Future Work

Based on informal observations, the system appears to detect motion more reliably when the moving object is closer to the robot and moving more rapidly. Because the robot's own motion is so rapid, it is understandable that slow-moving objects would be hard to detect. Near motion will appear more rapid in two dimensions; also, because of the downsampling, a moving object sufficiently far away will appear as a single pixel whose color is changing slightly.

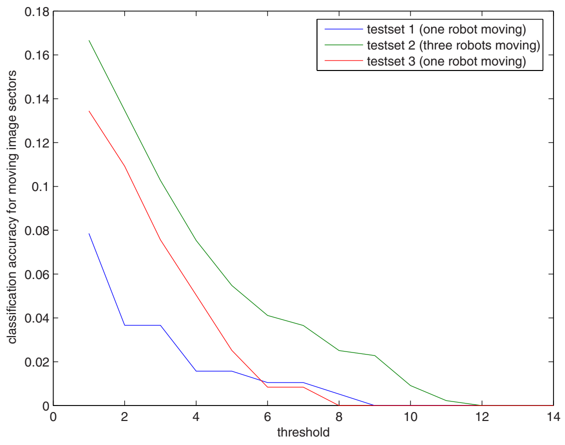

Fig. 6. Classification accuracy for image sectors containing motion. The system can detect a significant fraction of the motion in the robot's environment while maintaining the extremely low false positive rate shown in Figure 5. The x-axis corresponds to the number of edge pixels in a sector required for motion to be detected in that sector. As more edge pixels are required, fewer sectors containing motion are correctly labeled, leading to a tradeoff between accuracy in motion detection and elimination of false positives (Figure 5).

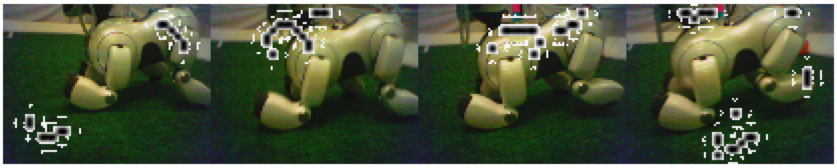

Fig. 7. Four sequential frames from a test run. The robot in the images is moving to the left at full speed. Overlaid black squares on the image indicate the discontinuities found by the postprocessing algorithm; when enough of these squares appear in a sector of the image, the system concludes that this sector contains motion.

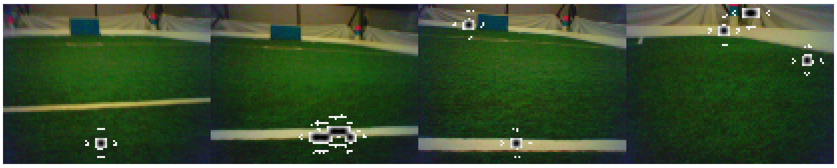

Fig. 8. Some typical errors on a field with no motion. The overlaid squares only indicate discontinuities, and enough of these squares must appear in the same sector for motion to be detected there. In the run from which these images were taken, in which there was no motion on the field, 83% of the frames contained no discontinuities of this type, and over 95% of the image sectors were correctly labeled as containing no motion.

This observation suggests an important direction for future work: extension of the system to work with larger images. Although the predictor neural network will have to be larger, it should not make training intractable: training time for the current network only requires a few hours of CPU time. Moreover, optic flow calculation over a half-resolution image (as opposed to the 1/6 resolution images currently used) has already been verified to be tractable.

Another important direction for future work is to implement the system to run in real time on an Aibo robot. This extension is plausible given the size of the images. The Aibo has a 576 MHz 64 bit RISC processor that allows for significant amounts of onboard computation. Although the computation required to process the images would reduce the rate at which images can be captured, a slower gait could compensate for any challenges posed to the optic flow algorithm by the increased time between images.

7 Conclusion

A method was presented for detecting external motion from a quadruped robot while it is walking freely and quickly. The system is resistant to false positives and is sufficiently accurate on real-world sensor data, and it is able to process this data with a speed that suggests that future onboard implementation is possible. Thus, it provides a way for a legged robot to sense motion in its environment, allowing it to direct its attention more intelligently, and ultimately making it more able to negotiate novel or changing environments.

Acknowledgments

Thanks to Mohan Sridharan for his help in 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 that supports this project. This research was supported in part by NSF EIA-0303609, NSF CAREER award IIS-0237699, and ONR YIP award N00014-04-1-0545.

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, IEEE Computer Society Press, Los Alamitos (2002)
4. Horn, B.K.P., Schunck, B.: Determining optical flow. Artificial Intelligence 17, 185–203 (1983)

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, 875–897 (1996)
6. Young, G.S., Hong, T.H., Herman, M., Yang, J.C.: Safe navigation for autonomous vehicles: a purposive and direct solution. In: Proceedings of SPIE: Intelligent Robots and Computer Vision XII: Active Vision and 3D Methods, vol. 2056, pp. 31–42 (1993)
7. Harris, C.L., Elman, J.L.: Representing variable information with simple recurrent networks. In: Proceedings of the 11th Annual Conference of the Cognitive Science Society, pp. 635–642 (1989)
8. RoboCup Technical Committee: Sony four legged robot football league rule book (2004), `http://www.tzi.de/~roefer/Rules2004/Rules2004.pdf`
9. Stone, P., Dresner, K., Fidelman, P., Jong, N.K., Kohl, N., Kuhlmann, G., Sridharan, M., Stronger, D.: The UT Austin Villa 2004 RoboCup four-legged team: Coming of age. Technical Report UT-AI-TR-04-313, The University of Texas at Austin, Department of Computer Sciences, AI Laboratory (2004)