In Gerhard Lakemeyer, Elizabeth Sklar, Domenico Sorenti, and Tomoichi Takahashi, editors, *RoboCup-2006*, Springer Verlag, 2007.

Selective Visual Attention for Object Detection on a Legged Robot

Daniel Stronger and Peter Stone
Department of Computer Sciences, The University of Texas at Austin
stronger,pstone@cs.utexas.edu
http://www.cs.utexas.edu/~{stronger,pstone}

Abstract. Autonomous robots can use a variety of sensors, such as sonar, laser range finders, and bump sensors, to sense their environments. Visual information from an onboard camera can provide particularly rich sensor data. However, processing all the pixels in every image, even with simple operations, can be computationally taxing for robots equipped with cameras of reasonable resolution and frame rate. This paper presents a novel method for a legged robot equipped with a camera to use selective visual attention to efficiently recognize objects in its environment. The resulting attention-based approach is fully implemented and validated on an Aibo ERS-7. It effectively processes incoming images 50 times faster than a baseline approach, with no significant difference in the efficacy of its object detection.

1 Introduction

Processing a stream of visual images is an important but time-consuming task. One technique that has been used to speed up vision processing is that of selective visual attention [1]. This technique is based on the idea that not all areas in a given visual scene are relevant to the task at hand. Therefore by restricting one's attention to the relevant parts of the scene, the agent can greatly increase its visual processing speed. This intuition is corroborated by work in cognitive science confirming that human vision processing takes advantage of selective attention. For example, Sprague et al. [2] present a model of visual attention and compare it to human eye-tracking data.

In robotic vision, selective attention can take two main forms. One is gaze direction, in which a robot moves its camera so that its field of view is faced towards the important information [3–5]. That approach is analogous to human eye saccading, but does not address the question of how to process each image, an often time-consuming process.

The other main approach to selective attention, which is taken in this paper, involves only processing the areas of the image that are likely to have relevant features. Because of the large amount of data in every image, processing each image in its entirety is difficult to do at frame rate, and where it is possible, it severely limits the amount of time the robot can spend processing the interesting areas of the image. By restricting its attention to the parts of the image that are most likely to contain the important information, the robot can dramatically speed up its image processing. This approach raises the challenge of identifying the useful areas of the image.

One common way to find the useful areas of the image is to first compute a saliency map [6], which represents the conspicuity at each point in the image. The most salient regions of the image can then be processed in detail. This approach has been applied to tasks such as face and handwritten digit recognition [7] and recognizing an object in a cluttered visual field [8]. Unfortunately, constructing the saliency map still requires processing the entire image, a time-consuming task in the context of trying to process a video stream at frame rate.

Another method for focusing attention on the important parts of the image is feature tracking [9, 10]. For example, Shi and Tomasi [9] present a method for identifying the optimal features in the image for tracking, i.e. the ones that are most likely to correspond to points in the real world. In feature tracking, a specific area of the image is analyzed between two consecutive images to characterize the movement of the feature. This method requires that corresponding points in consecutive images be close to each other, so that the tracking mechanism can properly identify the differences between consecutive frames as *movement*. However, there are sometimes cases in which objects in the field of view can move across a large portion of the image between consecutive frames. For example, in the case of a legged robot, the jerky motion caused by walking can lead to very sharp motion in the image. In these cases, feature tracking is not applicable.

This paper considers the situation of an autonomous legged robot equipped with a camera that moves within an environment with fixed landmarks. The robot's goal is to visually detect the landmarks as efficiently as possible. In this context, we present a novel technique for applying selective visual attention to the task of object detection. Like feature tracking, the technique is based on the idea that the robot can use prior information to predict the *expected location* in the image of each object. However, unlike previous work, it does not assume that an object's expected location in one frame is necessarily close to its location in the previous frame. The robot uses the objects' expected locations to direct its visual search towards the most likely areas of the image, enabling it to perform object detection very efficiently, despite the sharp motion caused by walking.

This approach presents three main challenges. First, predicting the location of an object in the image requires having an accurate estimate of the camera's *pose*, its position and orientation in space. On a legged robot, meeting this requirement is particularly challenging because as the robot walks, its body, and thus also its camera, rock quickly from side to side. Second, when preliminary analysis suggests that the object is not present at the expected location, the robot must have a strategy for continuing its search of the image for the target object. Third, the fact that most of the image is not processed at all presents a new challenge for object recognition: When processing a few pixels suggests the presence of an object, the robot must decide which pixels to process next to complete the object detection.

Although each image on our test platform contains over 33000 pixels, our technique allows the robot to examine fewer than 1200 of them on average as it identifies landmarks. This reduced processing enables the robot to process images 50 times faster than a baseline approach. Nonetheless, there is no significant drop in the success rate of object detection. The technique is implemented and validated on a popular mobile robot platform, the Sony Aibo ERS-7.

The remainder of this paper is organized as follows. The following section presents an overview of our technique. Sections 3-5 present solutions to the three challenges raised by this approach. Section 6 presents experimental results and Section 7 concludes and discusses future work.

2 Overview

In this paper we consider the task of a vision-based autonomous robot operating in a known environment. The robot has access to a series of images that are generated by a camera located on board the moving robot, which arrive at video frame rate. We assume that there is a fixed set of *objects* relevant to the robot's decision making (e.g., landmarks). The goal of the robot's vision module is to identify these objects when they are present in the image. The robot also maintains an estimate of its own pose in the environment over time, based on its visual observations and its odometry estimate. One popular approach to this self-localization problem, which we use in the experiments reported in this paper, is Monte-Carlo localization, or particle filtering [11, 12].

At each time step, the robot estimates its camera's pose. The details of how this is accomplished while the robot is walking are presented in Section 3. Then, given the camera's pose, the robot can loop through the fixed objects in the environment and, for each one, predict whether or not and where it is expected to appear in the robot's field of view. This prediction is achieved by projecting the object location onto the image plane (correcting for distortion if necessary).

For each object, if it is expected to be behind the image plane and therefore invisible to the robot, it is discarded. Similarly, if the object's projection onto the image plane is outside the field of view of the camera, it is discarded. Otherwise, the resulting image location is considered the object's expected location in the captured visual frame. If the expected location is examined and the object is not found, then the robot continues to search for it throughout the remainder of the image. This process is described in Section 4. Notably, a key challenge compared to previous approaches is that an object's location in the image plane may not be close to its location in the previous frame.

When preliminary analysis of an image location suggests that the object is present there, the robot must analyze that region of the image in detail. The goal of this processing is to accurately and efficiently determine whether or not the object is present in that location, and if so, how large it is in the image plane. A solution to this problem in our test-bed domain is presented in Section 5.

Finally, once the objects in the image have been identified, they can be used as landmarks for the purposes of localization. This process consists of converting the size and location of objects in the image into the corresponding distances and angles from the robot.

The entire method is summarized in Algorithm 1. The variable *Obj* loops through all of the environmental landmarks. Each one is projected onto the image plane if possible, initializing *TestLocation* to be the expected location of the object. This location is advanced by the routine *SeededSearch*, specified in Section 4. If the object is found, it is then used to inform localization. The underlined procedures in the algorithm will be described in the following sections.

Algorithm 1 Algorithm Summary

```
ComputeCameraPose
for all (objects Obj in the environment) do
  transform location of Obj into camera reference frame
  if (Obj is in front of the image plane) then
    project Obj onto ExpectedLocation in image plane
    if (Obj is in the camera's field of view) then
      TestLocation ← ExpectedLocation
      repeat
        Examine TestLocation for match with Obj
        advance TestLocation according to SeededSearch
      until (Obj is found) OR (entire image searched)
      if (Obj is found) then
        determine extent of Obj in the image
        project Obj back into global reference frame
        compute distance and angle from Obj
        incorporate information in localization
      end if
    end if
  end if
end for
```

3 Computing the Camera Pose

In order to accurately predict the location of the objects in the image, the robot needs to have an accurate estimate of its camera's pose. On a legged robot, this is particularly difficult because of the jagged motion caused by walking. The details of the method for finding the camera pose are necessarily dependent on the configuration and sensors of the specific robot being used. Nevertheless, the principles used here can be extended to apply to any robotic platform.

The experiments reported in this paper were performed on a Sony Aibo ERS-7.[1] The robot is roughly $280mm$ tall and $320mm$ long. It has 20 degrees of freedom: three in each of four legs, three in the neck, and five more in its ears, mouth, and tail. At the tip of its nose there is a CMOS color camera that captures images at 30 frames per second in YCbCr format. The images are 208×160 pixels giving the robot a field of view of $56.9°$ horizontally and $45.2°$ vertically. The robot's processing is performed entirely on-board on a 576 MHz processor.

Preliminary experiments have shown that an object's location in the robot's field of view can change by as much as 80 pixels between two consecutive video frames. This distance is a large fraction of the 208-pixel width of the image. The fact that a fixed point in the robot's view can move so far in one thirtieth of a second demonstrates the instability of the camera's pose. A video depicting the world from the robot's point of view is available online.[2]

The camera's pose can be estimated from the robot's joint angles and accelerometer values. This computation occurs in three phases. First, the height of the robot's body is estimated based on the back legs' joint angles. Second, the body's tilt and roll are estimated based on the accelerometer values. Finally, the head and neck angles are used to complete the computation.

[1] http://www.aibo.com

[2] `http://www.cs.utexas.edu/~AustinVilla/?p=research/selective-vision`

Ideally, the pose of the camera with respect to the ground plane could be computed by multiplying a series of homogeneous transformation matrices based only on the robot's joint configuration and angles [13]:

$$T_{foot}^{cam} = T_{foot}^{hip} \cdot T_{hip}^{body} \cdot T_{body}^{neck} \cdot T_{neck}^{cam} \tag{1}$$

where T_B^A represents the transformation from coordinate system A to coordinate system B. However, as the robot walks, the coordinate system of any given foot is not constrained to be either parallel to the ground or in contact with the ground. To compensate for this problem, we separately estimate the height of one of the rear hips and use the robot's internal accelerometers to estimate the body's tilt and roll. The resulting equation is

$$T_{ground}^{cam} = T_{ground}^{hip} \cdot T_{hip}^{body} \cdot T_{body}^{neck} \cdot T_{neck}^{cam} \tag{2}$$

In this equation, the *hip* coordinate frame is parallel to the ground. Then T_{ground}^{hip} is only a vertical translation, whose magnitude is determined by the rear legs' joint angles (based on the assumption that the more outstretched leg is the one touching the ground). The *body* coordinate frame is attached to the robot's body, so that the transformation T_{hip}^{body} must account for the body's tilt and roll. These quantities are estimated by using the robot's accelerometers. The three accelerometers report the component of gravity (combined with the body's acceleration) in the direction of each of the three cardinal axes in the body's reference frame. By taking the average of a rolling window of accelerometer values for each direction, the robot is able to filter out the effects of noise and acceleration and isolate the body's tilt and roll. The final transforms from the body to the neck and camera are done via standard DH-transforms, as described by Schilling [13]. The full details of our implementation are presented in our technical report [14].

Once the camera pose has been determined, it can be combined with the robot's prior body pose estimate in its environment to compute expected locations for objects in the robot's field of view. These locations can then be used to seed the visual search for those objects.

4 Seeded Visual Search

After the camera pose has been used to compute an object's expected location in the image, the robot must search for the object, as per Algorithm 1. This section describes our solution to the seeded visual search problem. That is, given that the robot has an expected location for an object in the image, how can the robot best take advantage of this knowledge to find the object as quickly as possible? We assume that, for each object, the robot has an *object decision mechanism* that can start at any pixel and eventually determine whether or not that pixel is part of the object in question. Ideally, the mechanism should usually reject pixels that are not part of the object after only some quick preliminary processing. For example, in a color-coded domain, the object decision mechanism can *segment* the pixel in question into a color category, and only for pixels that are the correct color, examine the surrounding region in more detail. In other settings the preliminary processing may require examining multiple pixels. The object decision mechanisms we use in our test-bed domain are described in Section 5.

Once the robot has identified an expected location for a given object, it first applies the object decision mechanism starting at the expected location. If the mechanism fails to find the object in its first application, the robot continues by applying the decision mechanism repeatedly, starting at a different pixel each time. If the mechanism starts at a pixel and successfully identifies the target object, the search can terminate. Otherwise, the searching process should continue starting the decision mechanism at different points, gradually moving outwards from the original expected location. In the worst case, the search could expand to cover the entire image. But in practice this rarely happens as will be demonstrated in Section 6.

To accomplish this goal, we use a series of starting points that follows a square spiraling pattern that expands outwards. The pattern used is depicted in Figure 1, and the points examined are all on a square lattice. The distance between lattice points depends on the minimum possible size of the target object. This property allows the robot to process as little of the image as possible while still ensuring that if the object is in the image it will be found. The spiral expands until either the target object is found or the entire image is filled. Because the image is filled in the worst case, the method is able to recover from a completely incorrect prior localization estimate.

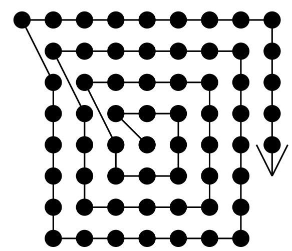

Fig. 1. The circles represent the locations in the image that are analyzed, starting from the expected object center. The distance between adjacent circles depends on the size of the target object.

5 Object Detection

The previous section describes how to process the image in a manner that enables the target object to be found quickly, given an object decision mechanism. This section presents our object decision mechanism, completing the description of the attention-based approach to object detection presented in this paper. First, however, it is necessary to specify the types of objects that are detected in the experiments reported in this paper.

The Aibo's environment used in this paper is a legged-league field measuring 4.4 meters by 2.9 meters. It has one blue goal, one yellow goal, and four visually distinct cylindrical beacons each with two colors: one is pink, and the other is blue or yellow. The robot is faced with the task of accurately identifying the goal and beacons that appear in the image.

Since the RoboCup field is color-coded, the object decision mechanism discussed in Section 4 analyzes pixels in the image by segmenting them and observing whether or not they are the color corresponding to the desired object: blue or yellow for a goal, or pink for a beacon. Color segmentation is carried out via a *color table*, a three-dimensional array with a color label for each possible combination of Y, Cb, and Cr values. The color table is created off-line, by manually labeling a large suite of training data and using a Nearest Neighbor learning algorithm to learn the best label for each YCbCr combination. The full details of our color segmenting algorithm are presented in our technical report [14]. After segmentation, a goal appears to the robot as a blue or yellow rectangle and a beacon appears as a pink square above or below a blue or yellow square of the same size. Whether the pink is above or below, and whether the other color is blue or yellow, uniquely determine which of the four beacons it is.

For each object, a single point is projected onto the image plane, as specified in Section 3, to yield an expected location. For goals, the center of the rectangle is used, while for beacons the center of the pink square is used. After the pink square is identified, the rest of the the beacon can be found easily.

As discussed in Section 4, an object decision mechanism is needed that examines the image starting at a given pixel and decides whether or not that pixel is part of the target object. The mechanism begins by segmenting the given pixel. If it is not the color corresponding to the target object, the mechanism rejects the pixel. Otherwise, the point is *expanded* into a maximal approximate rectangle of pixels all the same color. Since it starts with only one pixel segmented, it segments further pixels as they are needed.

To expand a point into a rectangle of a given color, we first expand the point into a line, by expanding to the right and then to the left. Expanding to the right consists of segmenting consecutive pixels, each one to the right of the previous one. This process continues until a sufficiently long string of consecutive pixels are all not the given color, marking the right end of the line. The threshold used for consecutive wrong-colored pixels, denoted as $ConsecThresh$, is three.

The pixel is expanded to the right and left, and then the process is repeated on the pixel directly above. This process continues upward until a pixel is found that cannot expand in either direction, which we consider to signify the top of the rectangle. Similarly the robot proceeds downward from the root pixel until the bottom of the rectangle is found. The rightmost right edge of an expansion line, and the leftmost left edge, are taken to be the left and right edges of the rectangle. Pseudocode for this point expansion routine is given in Algorithm 2. $BaseX$ and $BaseY$ are the image coordinates of the starting point for the expansion routine. The returned values represent the boundary coordinates of the rectangle.

Once a rectangle of the appropriate color has been found, the object decision mechanism decides whether or not the target object has been found, as opposed to some spurious pixels of the target color. For goals, the blue or yellow rectangle found is assumed to be the goal, as long as it is sufficiently large. For beacons, depending on the beacon's identity, the robot expects a blue or yellow square, either directly above or below the pink square that has been found. The location of the expected square center is based on the size and location of the pink square that has already been found. This new center is expanded into a blue or yellow rectangle in accordance with Algorithm 2. If this operation is successful and the resulting combined beacon rectangle is sufficiently large, the object decision mechanism registers a success. The resulting goal and beacon rectangles are the output of the vision processing.

Although the description in this section of the object detection mechanism is specific to our test-bed environment, the overall algorithm (presented in Section

Algorithm 2 Color Expanding Routine

Given: $ConsecThresh$, ColorTable, DesiredColor
Given: $BaseX$, $BaseY$
Given: ColorTable[Pixel[$BaseX$, $BaseY$]] = DesiredColor
$x \leftarrow BaseX$, $y \leftarrow BaseY$
repeat
  y goes up, then down, from BaseY, one pixel at a time
  repeat
    x goes right, then left, from BaseX, one pixel at a time
    Compare ColorTable[Pixel[x,y]] to DesiredColor
    if equal **then**
      ConsecutiveMisses $\leftarrow$ 0
    else
      ConsecutiveMisses $\leftarrow$ ConsecutiveMisses +1
    end if
  until on each side, ConsecutiveMisses reaches ConsecThresh OR x reaches image edge
until on top and bottom, y reaches a row where no DesiredColor is found or the image edge
$MinX$, $MaxX$, $MinY$, $MaxY$ $\leftarrow$ lowest and highest values of x and y, respectively to segment to DesiredColor
Return: $MinX$, $MaxX$, $MinY$, $MaxY$

2) does not depend on the details of the object detection mechanism. As long as the robot is equipped with a local method for determining whether a pixel or set of pixels depict an object of interest, that object decision mechanism can be plugged into Algorithm 1.

6 Experimental Validation

The goal of the method presented in this paper is to obtain a high object-detection accuracy without much computational complexity. As such, the mark of success is to perform roughly as well at object detection as a state-of-the-art approach that processes the entire image. In this section, we evaluate the technique presented in this paper by comparing it to a common baseline approach [14–16]. Section 6.1 describes our implementation of this baseline approach, and Section 6.2 presents experiments comparing the two methods.

6.1 Baseline Method

Our group has previously solved the problem of object detection on the Aibo ERS-7 on the RoboCup field using the baseline method mentioned above. However, this solution involves processing all of the pixels in every image, and despite the fact that we have optimized it aggressively, it consumes almost all of the robot's available processing time. The methods used are summarized in this section, while the full details of this baseline approach can be found in our technical report [14]. The robot executes the following three steps.

1. Color Segmentation: Classify every image pixel as one of a small set of distinct colors.
2. Region Merging: Collect adjacent pixels of the same color into monochromatic regions.
3. Object Detection: Identify the monochromatic regions that correspond to specific objects in the environment.

During the robot's image processing, it loops through every pixel in each image, classifying it according to the color table. Note that each image measures 208×160 pixels, for over 33000 pixels total.

As each pixel is segmented, it is also incorporated into a run-length encoding of the image. That is, each maximal horizontal string of consecutive pixels that are the same color is stored as a run-length. These run-lengths comprise a highly compressed version of the segmented image. Next, vertically adjacent run-lengths of the same color are combined into a *bounding box*, a rectangular structure consisting of the rectangle's coordinates and the color inside. The robot continues to merge bounding boxes that are adjacent and of the same color. Heuristics are used to determine if some adjacent boxes should not be merged, and also if some boxes should be deleted because they contain a density of the desired color that is too low. These bounding boxes are the input to object detection.

The robot first attempts to detect goals in the image, because they are generally the largest objects found. Thus the blue and yellow bounding boxes are sorted from largest to smallest, and tested for being the goal in that order. The goal tests consist of heuristics to ensure that the bounding box in question is close enough to the right height to width ratio, and that there is a high enough density of the desired color inside the bounding box for a goal, as well as other heuristics. To find the beacons, the robot finds bounding boxes of pink and blue or yellow that satisfy appropriate beacon heuristics, and then combines them into beacon bounding boxes, labeled by which beacon is inside. The resulting goal and beacon bounding boxes are the output of the vision process.

Once goals and beacons have been identified in the image, the robot uses them to update its localization estimate. This update takes place in two stages: translating the bounding box information into object distances and angles and incorporating these distances and angles into the robot's pose estimate.

The robot first computes its distance and horizontal angle to any objects it has identified. We have found it to be more effective to use the left and right edges of each goal instead of goal centers as landmarks for localization. To compute its distances and angles to the landmarks in the image, the robot takes into account the camera pose and the location and size of the object in the image. The distances and angles yielded by this process are often quite noisy, and it is rare to see enough objects in one frame to triangulate one's pose uniquely. To alleviate these factors, the robot maintains an estimate of its pose through particle filtering [11, 12], which gathers the robot's visual information and odometry into a coherent pose estimate over time.

The details in this section summarize our own baseline implementation for object detection that has been used successfully for a couple of years. Though there are many other object detection implementations within this domain that differ in their details [15–17], none of the approaches that we know of on the legged robot process the image based on the projected locations of objects.

6.2 Results

The baseline and attention-based methods were evaluated and compared based on their rate of success identifying landmarks in the image. In order to measure these success rates, we had the robot save a series of representative images to its memory stick with the identified landmarks marked. Then the images were viewed on a monitor and the objects were labeled manually.

In order to generate a series of representative images, the robot walked to a sequence of fixed poses on the field while continually moving its head from side to side, stopping at each pose for 15 seconds. The points and their order are shown in Figure 2. They were specifically chosen to represent a wide range of difficulty for object detection. Every ten seconds, the robot saved an image for evaluation. This ensured that a representative and varied sample of images was used for the evaluation.

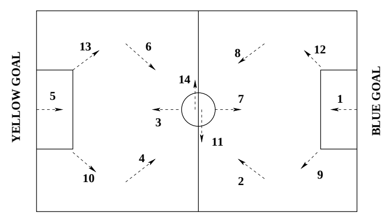

Fig. 2. The points on the field visited by the Aibo

Both methods were evaluated for ten trials. In each trial, the robot walked between the poses until it had taken 50 images total. The robot's rate of success within each trial was recorded. This rate is defined as the number of correctly identified landmarks (goal edges and beacons) divided by the number of landmarks identified by the manual labeling. This quantity represents the fraction of the objects that were actually in the image that the object detection successfully identified. We also counted the number of false positives: instances in which the robot reported a landmark that was not actually there. The false positive rate was extremely low for both methods. There were none at all recorded with the baseline method and two with the attention-based method over all ten trials. Compared to the total number of objects in the images captured while testing the attention-based method, 597, these errors represent a false positive rate of only 0.34 percent.

Furthermore, we recorded the amount of time taken to process each image. That is, for each image frame the robot receives, the amount of time taken to identify the vision objects in the image is recorded. These times do not include the time taken for particle filtering, nor for behavior and motion processing, which also occur in real time on-board the robot. The vision processing times are averaged over the full extent of all ten trials. We also measured the average number of pixels examined by the attention-based method on a typical run. This is compared to the number of pixels examined by the baseline method, which is the total number of pixels in the image. The average processing times, accuracies, and numbers of pixels processed attained by the baseline method and the new method are shown in Table 1.

The prediction-based method took an average of 0.695 ms to process each image, compared to the baseline method taking 35.049 ms on average. This represents a speed up of a factor of 50.4. To process at frame rate, there is 33 ms of computation time available per frame. Using the baseline method, vision processing takes 35 ms of computation, so that there is no time for additional processing while continuing to operate at roughly 30 Hz. In fact, even with minimal computation performed for localization and decision-making, the baseline process leads to an ability to process at only 24 Hz. Thus the robot completely ignores information from 20% of the available frames. In contrast, when using our method, the vision processing of stationary objects takes only 0.7 ms, leaving more than enough time to perform localization and decision-making while operating at 30Hz, and in fact freeing up much additional processing time. This time could potentially be used for precise tracking of mobile objects, as well as enabling the exploration of completely new approaches that would not have been tractable previously, perhaps such as detailed behavior planning or teammate/opponent modeling.

Method	**Baseline**	**New Method**
Avg Detection Rate	$\mathbf{77.54 \pm 7.32}$	$\mathbf{74.33 \pm 10.63}$
Avg Pixel Count	**33280**	**1138**
Avg Time/Frame	**35.049** ms	**0.695** ms

Table 1. Comparison of the baseline and new, prediction-based methods.

At the same time, the new method achieved a very similar quality of vision accuracy to the baseline method. Over the ten trials, the attention-based method attained an average detection rate of 77.54 ± 7.32 percent, while the baseline method attained an average rate of 74.33 ± 10.63. Based on the variances in these measurements, the difference in their means is statistically insignificant ($p > 0.4$ in a two-tailed t-test). These tests demonstrate that the attention-based method sped up the robot's vision processing by a factor of 50, without any significant effect on the success rate of object detection.

7 Conclusion and Future Work

This paper presents a technique for a legged robot equipped with a camera to use visual selective attention to efficiently recognize objects in its environment. Given the inapplicability to this domain of previous techniques, a novel approach is presented based on taking into account the robot's prior knowledge about its camera's pose. This approach presents three general challenges, which are particularly difficult in the context of a legged robot. The paper presents solutions to each of those challenges in that context.

When these solutions are combined, the resulting technique enables a legged robot to process its visual data stream very efficiently. We have fully implemented and validated this technique on an Aibo ERS-7. This attention-based approach effectively processes incoming images 50 times faster than a baseline approach, with no significant difference in the efficacy of its object detection.

This work has a number of interesting possibilities for future extensions. One enhancement would be to extend the method so that it could also find mobile objects. Achieving this goal would require having the ability to predict the objects' locations in advance. Such an ability could derive from a predictive model of the objects' locations based on their velocities, communication with other robots, or a combination thereof. Another possibility for future work would be for the robot to draw cues from objects that have already been processed in a given image to decide which regions of the same image will be most likely to contain other objects. Furthermore, these methods could be extended so that they influence gaze direction as well as the focus of attention within each image. Finally, this technique is designed to be generally applicable and can therefore be applied in different domains to ascertain the range of possible environments and types of objects with which it can be effective.

Using the camera's pose to speed up vision is an example of enhancing vision processing by incorporating high-level information. This general idea is promising, based on the notion that in order to effectively perceive its environment, an agent should take advantage of as much prior knowledge as possible about that environment. This work represents progress towards the long-term goal of enabling autonomous agents to make effective use of their knowledge of the world in their perceptual processing.

Acknowledgements

This research was supported in part by NSF CAREER award IIS-0237699, ONR YIP award N00014-04-1-0545, and DARPA grant HR0011-04-1-0035. The authors thank Mohan Sridharan and Aniket Murarka for helpful suggestions.

References

1. A. L. Yarbus, "Eye movements during perception of complex objects," in *Eye movements and vision*, L. A. Riggs, Ed. New York: Plenum Press, 1967, ch. VII, pp. 171–196.
2. N. Sprague, D. Ballard, and A. Robinson, "Modeling attention with embodied visual behaviors," 2005, http://www.cs.rochester.edu/~dana/WalterTheory25.pdf.
3. N. Mitsunaga and M. Asada, "Sensor space segmentation for visual attention control of a mobile robot based on information criterion," in *Proceedings of the IEEE International Conference on Intellegent Robots and Systems*, 2001.
4. C. Kwok and D. Fox, "Reinforcement learning for sensing strategies," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2004.
5. J. Najemnik and W. Geisler, "Optimal eye movement strategies in visual search," *Nature*, vol. 434, pp. 387 – 391, 2005.
6. L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, Nov 1998.
7. A. A. Salah, E. Alpaydin, and L. Akarun, "A selective attention-based method for visual pattern recognition with application to handwritten digit recognition and face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, March 2002.
8. D. Walther, U. Rutishauser, C. Koch, and P. Perona, "On the usefulness of attention for object recognition," in *The 2nd Workshop on Attention and Performance in Computer Vision*, 2004.
9. J. Shi and C. Tomasi, "Good features to track," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 1994.
10. S. Baluja and D. Pomerleau, "Expectation-based selective attention for visual monitoring and control of a robot vehicle," *Robotics and Autonomous Systems*, vol. 22, no. 3-4, 1997.

11. F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1999.
12. M. Sridharan, G. Kuhlmann, and P. Stone, "Practical vision-based monte carlo localization on a legged robot," in *IEEE International Conference on Robotics and Automation*, April 2005.
13. R. Schilling, *Fundamentals of Robotics: Analysis and Control*. Prentice Hall, 2000.
14. P. Stone, K. Dresner, P. Fidelman, N. K. Jong, N. Kohl, G. Kuhlmann, M. Sridharan, and D. Stronger, "The UT Austin Villa 2004 RoboCup four-legged team: Coming of age," The University of Texas at Austin, Department of Computer Sciences, AI Laboratory, Tech. Rep. UT-AI-TR-04-313, October 2004.
15. J. Bunting, S. Chalup, M. Freeston, W. McMahan, R. Middleton, C. Murch, M. Quinlan, C. Seysener, and G. Shanks, "Return of the NUbots! the 2003 NUbots team report," 2003, http://robots.newcastle.edu.au/publications/NUbotFinalReport2003.pdf.
16. N. Mitsunaga, H. Toichi, T. Izumi, and M. Asada, "Babytigers 2003: Osaka legged robot team," 2003, http://www.er.ams.eng.osaka-u.ac.jp/robocup/BabyTigers/BabyTigers-TechR%eport-2003.pdf.
17. T. Roefer et al., "German team: Robocup 2004," 2004, http://www.germanteam.org/GT2004.pdf.