Real-Time Object Detection using Segmented and Grayscale Images

Juan Fasola and Manuela Veloso

Carnegie Mellon University
Computer Science Department
5000 Forbes Avenue, Pittsburgh, PA, USA, 15213

Email: jfasola@andrew.cmu.edu, veloso@cs.cmu.edu

Abstract - **This paper describes an approach that performs visual object detection in real-time by combining the strength of processing the color segmented image along with that of the grayscale image of the same scene. This approach was developed with the annual RoboCup[1] Competition in mind, specifically the 4-Legged League where teams of Sony AIBO robots compete in the game of soccer [2]. The images used for processing were taken from the camera located in the head of the robots, and the objects of interest to be detected were the actual AIBO robots. We use color segmented images for producing initial hypotheses for the location of robots in the image, and grayscale images for final classification purposes. Using both representations to process a scene allows each to make up for the deficiencies of the other, and provides a good balance between fast processing time and high detection accuracy. We present our algorithms and show illustrative examples of their performance.**

Index Terms – Real-Time, Object Detection, Vision.

I. Introduction

In RoboCup, the detection of teammate and opponent robots is an interesting research problem, as the information gathered is useful in a variety of different areas, such as obstacle avoidance, teammate coordination, active strategy, and world modeling. For example, a robot may decide to play a more defensive role if it detects that an opponent robot has control of the ball. Furthermore, the detection of other robots introduces new characteristics to the world state, such as the concept of occlusion.

In AIBO robot soccer, the robots have a very limited field of view with the camera on their highly movable head. The soccer playing robots are not able to constantly see the ball for a variety of reasons, including (i) they need to track multiple objects in the field, namely the lines, the markers, and the goals; (ii) the other robots are tall enough to hide the small ball; and (iii) the position of the camera on one robot's head also disables seeing the ball when the ball is too close to the robot's own body – such a ball position is needed in order for the robot to actuate (e.g., kick) over the ball. Therefore our world state modeling algorithm [3] needs to robustly reason about the multiple times that the robot does not see the ball and predict, as accurately as possible, where the ball should be. If the robot does not see the ball in the image, the world modeling algorithm could assume that the ball is just not there. However that may not be the case, as we explained above. Our detection of other robots, as presented in this paper, allows the world modeling algorithm to correctly predict that the ball is being occluded by a robot, if indeed a robot is detected in a conflicting position with the expected ball position. This feature has the potential to greatly improve the navigation of the robot and its search for the ball.

II. Robot Object Definition and Approach Overview

The AIBO robots are white, in the shape of a small dog with four legs and a moving head. In a robot soccer game, the robots wear special "uniforms," which are colored patches of different (strange) shapes that cover parts but not all of their white body. The patches are either red or blue to denote the team but all the robots in a team are indistinguishable.

One way to address the problem of robot detection is to search for red or blue blobs in the image, and if the blobs conform to some constraints, a robot is detected. In theory, this seems to be a reasonable solution, however the color of the blue uniforms is so dark that in the color segmented images they appear to be black. Since the background color is usually black, and there are a lot of shadows and segmentation noise that is also black, finding blue robots in the image with this technique is not as easy as it seems. If one were to try and correct the segmentation to emphasize the recognition of the color blue of the uniforms, then a lot of black pixels would become blue, and the problem would still remain. Also, there is more than one uniform patch on any given robot, and so determining whether or not blobs are part of the same robot or multiple robots correctly is non-trivial. Furthermore, the uniforms are only one aspect of the robots, and intuitively it seems that by using only the uniform color information, much information is ignored that would otherwise help to greatly improve the detection of robots. For these reasons we have decided not to use uniform colored blobs for robot detection, but focus rather on finding objects located on the playing field and afterwards determine whether or not the objects found are robots by comparing them to pre-determined models of what robots look like from various views.

Our approach involves processing both the color segmented image and grayscale image taken by the robot of a given scene on the playing field. The segmented image is used for rapidly finding objects that are lying on the field, which become the initial hypotheses of robot locations. The grayscale image is then used to analyse the area of the hypotheses more closely to classify whether or not the objects detected are robots. The details on how this classification is determined are explained in the next section. The segmented image can be thought of as providing the areas of interest to the grayscale image classifier, which greatly simplifies its task of robot detection by minimizing the search space. The segmented image, by virtue of its color information, has the advantage over the grayscale image of being able to find objects located on the playing field very rapidly, whereas the grayscale image, by virtue of its level of detail, can provide a much more accurate classifier than the segmented image. Thus, both image representations are used for their strengths and to make up for the deficiencies of the other. The result is an object detector that yields higher performance, when taking into account speed and accuracy, than when only one or the other image representation is used.

III. Robot Detection Algorithm

A. *Segmented Image Hypothesis*

The first stage of the approach is to find the initial hypotheses of robots located in the image. These location hypotheses are computed very rapidly by using the color segmented image available to us for the current vision frame. The color segmented image is created by applying the CMVision[4,5] color segmentation algorithm to the YUV image provided by the hardware of the Sony AIBO for each frame. The initial hypotheses that are computed essentially give the locations of any type of object or obstacle on the playing field. All the images that need to be processed come from the head of an AIBO that is standing on the field, therefore certain assumptions about the image are made to aid and speed up the detection of objects that are considered to be on the field. There are three main assumptions:

1) Field horizon: There exists a horizontal line that best separates the playing field from everything above it, called the field horizon. Every pixel below this horizon line belongs to either part of the field or something on the field.

2) Objects not green: Objects on the field, such as robots, people's legs and other things, are not green in color since the field is green and a distinction must be made between objects on the field and the field itself. Therefore, every pixel below the field horizon that is not green is part of an object that is located on the field.

3) Objects intersect field horizon: Objects standing on the field will always intersect the field horizon. This assumption is made because the robots to be detected will most likely always be standing, and from the point of view of another standing robot the playing field cannot be seen over the robot being viewed.

By using these assumptions the initial hypotheses of robots on the field can be found much more rapidly than if these assumptions were excluded.

Our color segmented image object detection algorithm generates hypotheses on the position of robots in two main steps, namely the detection of the field horizon, and the detection of non-field, i.e., non-green, present objects. Fig. 1 shows pseudo-code for this object detection algorithm.

```
Algorithm I: GETHYPOTHESISWINDOWS(image)

min ← ∞
RESCALEIMAGE(image)
for each column col in image
   row ← FINDSTARTOFGREEN(image, col)
   if row < min
      min ← row
fieldHorizon ← min
for each column col in image
   objs_col ← FINDOBJECTEND(image, fieldHorizon, col)
return FORMROBOTWINDOWS(fieldHorizon, objs)
```

Fig. 1 Algorithm for creating initial hypothesis windows for robot locations within image frame.

The field horizon is the first feature that needs to be detected, as subsequent processing, namely the detection of object regions, relies on it. The detection algorithm as of now handles only the case when the camera is minimally rotated, therefore the field horizon is assumed to be a horizontal line. The field horizon is calculated by first rescaling the initial 208x160 image to a resolution a factor of four smaller along each dimension, which is 52x40, by using a nearest-neighbor approach. At this resolution, we can scan along the image columns sequentially in order to find the highest point in the image where the number of consecutive green pixels detected exceeds some threshold, in our case four pixels. The four green pixels are assumed to be part of the green colored field, and the highest point in the image where this sequence occurs defines the field horizon line.

Objects located on the field are assumed to be below the field horizon and have some color other than green that differentiates them from the field. The method for finding these objects is to scan along each column of the smaller resolution image, starting from below the field horizon, to search for non-green pixels. If enough green is detected in sequence along a column the field is considered reached, which signals the end of the object being scanned. There is no need to continue searching down the column for object colored pixels after this as objects are assumed to intersect the field horizon. "Enough green" is dependent upon how much green is encountered and at what height in the image, in order to accommodate cases where parts of the field are seen through a robot's legs. For example, one green pixel immediately below the field horizon is enough to stop a column scan, however further away from the field horizon sequences of two or three green pixels may not be enough to stop a column scan. This procedure essentially finds the length along each column of an obstacle that intersects the field horizon line. Neighboring columns with lengths a reasonable amount apart are grouped together. The lengths of all the columns within a group are averaged to find the length for that group. Using this length, along with the starting and ending column positions, a bounding box is created for each group; these bounding boxes represent the initial hypothesis windows for robots located on the field. Fig. 2 shows illustrative results of our color segmented image object detection algorithm.

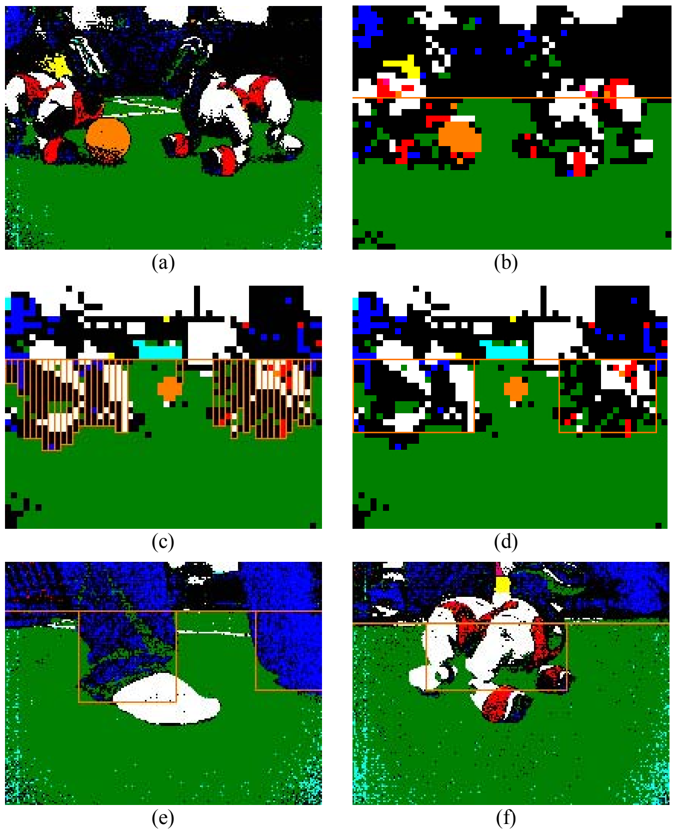


Fig. 2 Illustrative procedures and example results of the color segmented image object detection method. (a) Initial color segmented image. (b) Reduced image with detected field horizon. (c) Initial column scan results for object colors. (d) Final detected objects after grouping columns. (e) Detection result for a person's legs. (f) Detection result for an AIBO on the field.

B. *Object Classification using Grayscale Image*

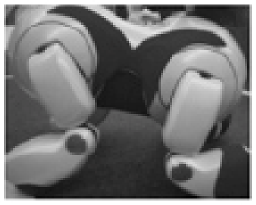
The initial hypotheses calculated using the color segmented images essentially represent the location of all objects on the field. Therefore, we use a classifier to determine whether or not the detected objects are actually robots. The classification makes use of the grayscale image of the current vision frame, which is easily inferred from the YUV image returned by the robot hardware. One key component of the classification method is the "integral" image which was originally introduced by Viola and Jones in order to perform real-time face detection [6]. The integral image can be computed in one pass over the grayscale image, and allows the sum of pixel intensities for any rectangular region in the image to be computed in constant time. This, in turn, allows the mean pixel intensity of any rectangular region in the image to be calculated in constant time. Our algorithm makes use of this feature of the integral image to speed up detection time.

The idea behind our classification method is to store some image that represents a model of what robots look like, and later compare the image windows of the robot hypotheses to this model for classification purposes. The model is compared to the hypothesis windows through features. Fig. 3 shows the pseudo-code of our grayscale image based classification algorithm.

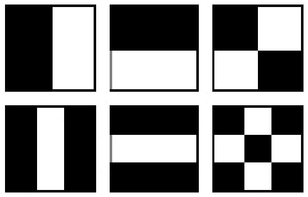
```
Algorithm II: ROBOTINSIDEWINDOW(window)

for each model view image $model_i$
   $robotFound \leftarrow$ true
   for each mask $mask_j$
      $f_m \leftarrow$ CALCMASKFEATURE($model_i$, $mask_j$)
      $f_w \leftarrow$ CALCMASKFEATURE($window$, $mask_j$)
      $d \leftarrow$ EUCLIDEANDISTANCE($f_m$, $f_w$)
      if $d >$ THRESHOLD
         $robotFound \leftarrow$ false
         break
   if $robotFound$
      return true
return false
```

Fig. 3 Algorithm for classifying whether or not robots are contained within the given grayscale image window.

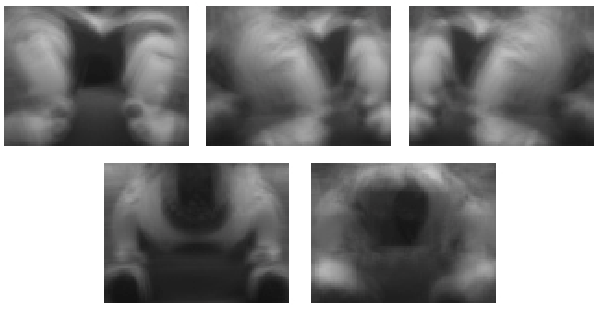
A feature is a collection of mean pixel intensities of rectangular regions in the image defined by a mask. The mean pixel intensities of the features are rapidly computed by using the integral image representation described above. The model is represented by a collection of six different features, each with its own mask (Fig. 4b). Each robot hypothesis window is compared to the model by having its own six features computed and subsequently calculating the Euclidean distance between each of its features with the model features and applying a threshold to the result. The comparison is performed in a cascade, that is, if one of the model features is not sufficiently close to the corresponding feature of the hypothesis window, the window is classified as a non-robot and no further processing is done on that hypothesis window. Therefore, in order for a hypothesis window to be classified as a robot, all the model features must be sufficiently close to the corresponding features of the window, defined by a hand-tuned threshold. Now, instead of having only one model of the robot, we have a total of five models (Fig. 4c). When analysing a hypothesis window all models are considered separately, however if one model classifies the window as a robot then no further processing is done on that hypothesis window. Each of the model images used represent one possible view of the robot. The five views that we chose for the robot model images are: side, angled right, angled left, front, and back. Having multiple models of the robot from different viewpoints increases the overall robustness of the detector by improving its accuracy and decreasing the number of false negatives it produces. For each of the five different views, around thirty images were taken (Fig. 4a) and their mean image was used as the model image for robots at that view.



(a)



(b)



(c)

Fig. 4 (a) An example side view image used when creating the mean image for the side view model. (b) The six feature masks defining the rectangular regions whose mean pixel intensities were used to create features that enabled comparison between the model images and hypothesis windows. (c) The five robot models used in the approach, each representing a possible view of the robot, which are: side, angled right, angled left, front, and back.

C. *Combining Both Methods*

Integrating both detection methods described to form one final robot detection method is the last stage in our approach. The first step in combining both methods is to just run the first method as-is. The color segmented image processing provides us with initial hypotheses of where robots are located in the image. Given these hypothesis windows, which are generally very tightly fit to the objects, we expand them by a few pixels to allow the grayscale image classifier more room to search for robots, thus improving the detection rate. Each enlarged hypothesis window is then considered to be an image in itself, and many sub-windows within the hypothesis window are classified.

Classification is greedy in the sense that if one sub-window is classified as a robot, then the whole hypothesis window is classified as a robot and no further processing is done on that hypothesis window. A hypothesis window is classified as a non-robot if every sub-window considered is classified as a non-robot. The first sub-window classified for a given hypothesis window is exactly the hypothesis window, since it is assumed to be the most descriptive sub-window, and hence the most reliable to classify the window as containing a robot immediately, which would minimize the amount of processing done on the hypothesis window. If the first sub-window fails to find a robot, more sub-windows are checked for robots.

The remaining sub-windows that are considered for classification are windows of different scales, starting with the base scale defined by the programmer, and increased by a scaling factor until the maximum scale window is reached that fits within the hypothesis window in question. At each scale, the sub-window being considered is moved across the entire hypothesis window, starting from the top-left corner on down. The sub-window is slid across the hypothesis window a distance of DX*[current scale] pixels per movement along the x direction, and a distance of DY*[current scale] pixels per movement along the y direction. This way, larger scale sub-windows move along the hypothesis window at bigger steps then smaller sub-windows. Fig. 5 presents pseudo-code for the algorithm that combines both detection methods.

```
Algorithm III: FINDROBOTS(colorimage, grayimage)

HypW ← GETHYPOTHESISWINDOWS(colorimage)
for each window $HypW_i$
    W ← GETGRAYSCALEWINDOW(grayimage, $HypW_i$)
    if ROBOTINSIDEWINDOW(W)
        $R_i$ ← true
        continue
    $R_i$ ← false
    for scale s within W and not $R_i$ ;  s ← s * SCALEFACTOR
        for y within W and not $R_i$ ;  y ← y + DY * s
            for x within W and not $R_i$ ;  x ← x + DX * s
                subwindow ← GETSUBWINDOW(W,x,y,s)
                if ROBOTINSIDEWINDOW(subwindow)
                    $R_i$ ← true
return R
```

Fig. 5 Algorithm for finding robots within image frame, which combines the color segmented image object detection method with the grayscale image robot classifier.

In summary, for every initial hypothesis window returned by the first method, a search is performed to detect robots within the window by first classifying the whole window, then classifying sub-windows moved across the window which increase in scale until reaching the maximum scale allowable within the hypothesis window. Sub-windows are classified until one has been classified as a robot, or until all possible sub-windows have been classified. The number of possible sub-windows depends on the size of the initial hypothesis window, the scaling factor, and the movement parameters DX and DY. Most of the time, sub-windows that do not contain robots are classified as such very rapidly due to the cascade of comparisons described in section B, therefore classifying all possible sub-windows is not as time consuming as it may seem.

IV. RESULTS

To test the final combined robot detector a set of 327 test images were classified. Using a scaling factor of 1.25, and step size parameters DX and DY of 1, the final detector was able to achieve a 97% classification accuracy on the test set, yielding only one false positive, with the remaining false classifications being false negatives. The false negatives were from the harder to classify views of the front and the back. It is possible to reduce the number of the false negatives by fine-tuning the thresholds for model/hypothesis comparison, however at the expense of increasing the number of false positives. False positive robot classifications are considered worse than false negatives, therefore a balance was achieved between the two by fine-tuning the thresholds by hand. Fig. 6 provides examples of these two failure cases.

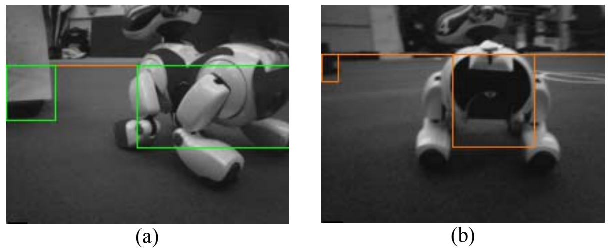
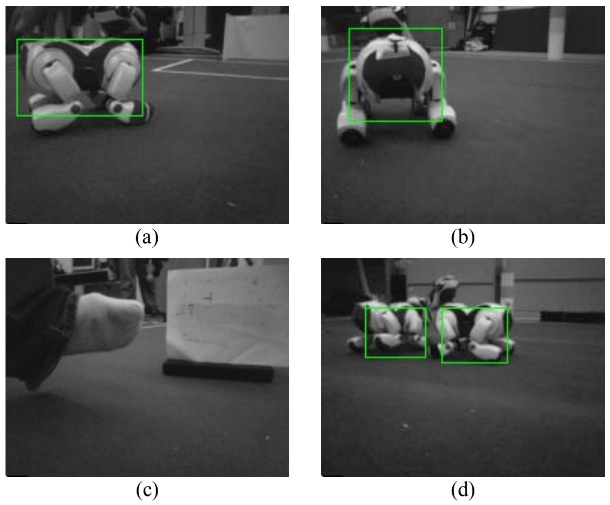


Fig. 6 Failure cases with combined detector. (a) False positive on the left. (b) False negative on robot, with initial hypothesis shown.

The speed of the detector for any given frame depends on the number of initial robot hypotheses and the number of sub-windows analyzed per hypothesis. The detector was tested on a 3 GHz Pentium 4 processor and was able to achieve speeds higher than 60 fps for many images in the test set, however it sometimes fell below 10 fps, with the lowest recorded speed of 5 fps. On average, the detector evaluated the test set at 30 fps, which achieves our goal of real-time visual image processing. Fig. 7 shows some example results of our robot detection algorithm.

One limitation of the algorithm, as it is comparison based, is that robots that are occluded or lie on the edge of the image are not classified as robots because their full image is not in view. While this is unfortunate, it is trade-off between having to account for occlusions and most likely causing more false positives and slower processing time.



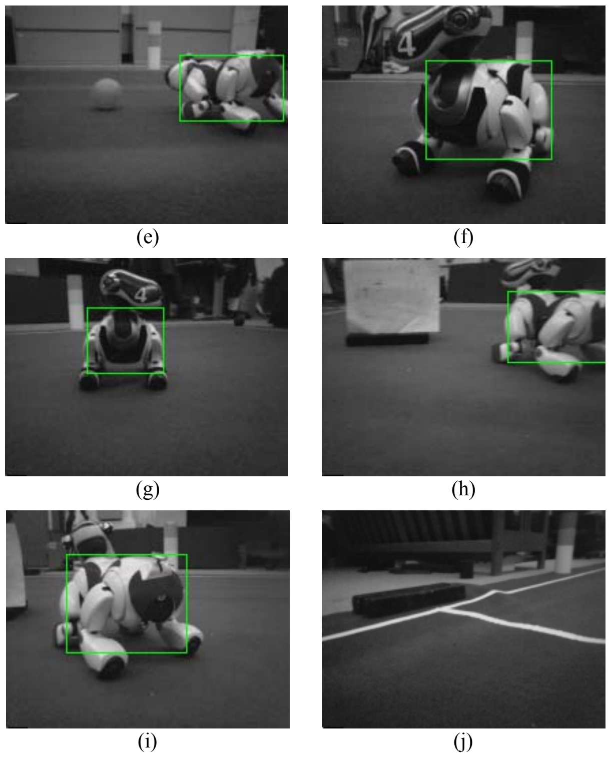


Fig. 7 Results of the combined robot detector which combines the color segmented image object detection method with the grayscale image robot classifier

Our detection algorithm performs very well even when presented with robots in the image that do not fall exactly into one of the five view categories that we've described. For example, in Fig. 7f the robot in the image seems to be oriented in a manner between a frontal view and an angled view to the right, yet our detector is able to handle this case. Fig. 7i shows a similar situation, except that the robot in the image is oriented away from the viewpoint. Our detector is also able to detect robots even when they are not completely standing upright or are in an unusual configuration. Fig. 7e shows exactly one of these interesting situations, one in which the robot is in the middle of head kicking the ball and has lowered its body closer to the ground so as to hit the ball more effectively. This demonstrates the capability of our detector to use a limited set of robot model images and generalize over a larger domain of robot configuration images.

V. Conclusions

We have presented an algorithm that performs visual object detection by using both the color segmented image and grayscale image of the same scene to achieve real-time processing rates. The approach was used to detect robots in images, in the context of RoboCup, however the approach can be easily be extended to detect a variety of different objects, given models that can effectively discriminate between the desired object and other objects in the image.

REFERENCES

[1] H. Kitano, Y. Kuyinoshi, I. Noda, M. Asada, H. Matsubara, and E. Osawa. RoboCup: A challenge problem for AI. *AI Magazine*, 18(1), pages 73-85, 1997.

[2] M. Fujita, M. Veloso, W. Uther, M. Asada, H. Kitano, V. Hugel, P. Bonnin, J.-C. Bouramoue, and P. Blazevic. Vision, strategy, and localization using the Sony legged robots at RoboCup-98. *AI Magazine*, 1999.

[3] Maayan Roth, Douglas Vail, and Manuela Veloso. A real-time world model for multi-robot teams with high-latency communication. In *In Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2494–2499, Las Vegas, NV, October 2004.

[4] J. Bruce, T. Balch, and M. Veloso, "Fast and inexpensive color image segmentation for interactive robots," *in Proc. of IROS-2000*, Japan, October 2000.

[5] CMVision web page *http://www.cs.cmu.edu/~jbruce/cmvision*

[6] P.Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *in Proc. CVPR*, Kauai, HI, 2001.