

Using pattern matching on a flexible, horizon-aligned grid for robotic vision

Joscha Bach, Matthias Jünger

Humboldt University of Berlin
Department of Computer Science, Artificial Intelligence
bach|juengel@informatik.hu-berlin.de

Abstract. Image processing in the robotic soccer domain meets real time constraints that must be met with optimizations of the task. We are proposing a method to find objects using a non-uniformly spaced, optimally constructed grid. We are capitalizing on the fact that the apparent size of ground-based objects is related to their visible distance to the horizon. In a three-stage process, a pattern matching is performed along the grid-positions, object candidates are identified by specialized modules, and finally, the consistency of the result is improved. We describe an example identifier module in more detail.

1 Introduction

Building and testing embodied agents – robots – confronts AI research with many situations, where constraints imposed by the complex environment of the real world call for solutions similar to those in biological organisms.

An example is the Sony four-legged robot league of robotic soccer [RoboCup], which provides a number of very interesting and demanding challenges, among them very specific demands of computer vision. Unlike in many other robotic applications, it is impossible to adapt sensors and processing resources to the task of recognizing the environment, for instance, it is not possible to augment spacial recognition with laser or sonar sensors. Rather, the used algorithms have to cope with the features of the hardware provided, which includes a CCD camera with a fixed lens and a resolution of 176 by 144 pixels.

A commonly used method to process these camera images consists of a simple three step strategy: First the color class of each pixel in the image is determined using a look-up table. Second, neighboring pixels in the color classified image are combined to regions. Finally, characteristic properties of the regions like size, area, bounding-box etc. are used to identify and locate the objects of interest in the image. The first step usually needs a significant fraction of the time, because every single pixel of the image is accessed. The time necessary for the other phases depends largely on the nature of the image. [Bruce, Balch, Veloso 00]

It is possible to speed up the processing by using a low-resolution version of the image, but the trade-off for this consists in a loss of accuracy and correctness. However, there are better strategies.

The objects of interest in an image cover usually only a fraction of the image. Moreover, not all interesting objects need to be processed with the same level of detail. Thus, the image processing task may consist of several stages:

- Find regions of interest with fast, simple methods and direct the attention to it
- Verify objects in these regions and identify them with specific methods
- Improve the overall consistency of the result and use it for interpreting the perceived environment

This is somewhat similar to the human vision system, which is able to react quickly to simple, so called 'pop out' properties to direct its attention [Triesman 85]. Methods to capitalize on this for image processing have been described for instance by Kaelbling et al. [2001].

Because objects of interest often span over relatively large areas, it is usually not necessary to check every single pixel to find them. If it is possible to locate them roughly with a low-resolution search and if the edge of the objects is characterized by features like a color or lightness change, the high-resolution image can be traversed very efficiently in a way that follows mostly just the outlines of objects and locates them with maximal accuracy. [Quek 00]

The standard method to scan an image with a lower resolution is the use of a grid. While objects close to the observer tend to appear very big and require only a low resolution grid to spot them, remote objects often have the unfortunate tendency to shrink down to a few pixels in diameter. If it is known where to look for close and where to look for remote objects, a non-uniform grid could be used [Kierzenkowski, Puchalski 94]. Fortunately, we have such knowledge, because most objects of interest are located at ground level (with the exception of the flags that mark the borders of the field). Their distance can be estimated by their position relatively to the horizon (depending on the height of the camera).

Our robots have some knowledge about the orientation of their camera, which we use to construct a horizon-aligned perspective distorted grid. We also handle objects that are suspended above ground level (i.e. flags). We demonstrate how using such a grid enables the usage of simple pattern matching strategies for finding candidates for object detection. In the last stage, the grid-positions of these candidate objects are given to specialized algorithms that try to detect them and find their outlines. While we have developed our method for using it for our team in the Sony four-legged robot league [GermanTeam], it may also be useful for other robotic leagues or applications outside the soccer domain.

The paper is structured as follows: The next section describes the criteria for the construction of the grid. Section 3 covers a simple, yet efficient one-dimensional pattern matching process for finding object candidates. Section 4 explains an example object identification module: a ball identifier.

2 Using a horizon-aligned distorted grid

Consider an object of fixed size and an image of that object taken by a robot's camera: then the size of the object in the image depends linearly on its distance to the camera. If the task is to find a structure of a certain size in an image and every pixel is characteristic for that structure, it is sufficient to have a look at pixels positioned on a grid with a spacing that is more narrow than the size of the structure.

In the worst case, if objects are far away from the agent, the grid can condense to a width of just a single pixel. By limiting the minimal grid size, agents will not find reliably small objects that are farther away than a certain distance.

Fortunately, like many robot applications, the setting of our robots is ground based, i.e. objects touch the ground and are usually characterized by features closely located to the ground. Therefore, there is also a relationship between the distance and the position of objects: those that are far away are close to the horizon. If the grid narrows toward the horizon, it can make use of this property. Note that the camera height is important too: if the camera is close to the ground, the minimal size of objects closely located to the footing point of the camera is relatively big, while a high vantage point makes for smaller object appearances.

These observations give us two parameters determining the structure of an optimal grid that covers ground based objects independently of their position: camera height and camera rotation. To work out how these parameters influence the structure of the grid imagine an arrangement of a set of equal objects on the ground, such that the images of the single objects intersect but do not overlap in the image taken by the camera. Naturally, the shape of the objects will influence the grid; if we are looking for post-cards laid out on the ground, the optimal grid will have other properties than in the case of spherical objects.

For our application, we choose spherical objects (they are well suited for abstracting object sizes, and coincidentally, the smallest object we are interested in happens to be a ball). We want to make sure that every ball that can lie on the ground plane up to a certain distance is covered at least by one grid-point, while the grid consists of horizontal lines (not aligned with the image, but parallel to the horizon) and intersecting lines running towards a vanishing point.

We may distinguish the following properties:

- The grid can be *line covering*, that is, for each possible position of the object in the image at least one line of the grid intersects with the object. (Imagine the grid to act like a sieve where the lines are acting like wires. A line covering grid will hold back all objects.)
- A grid can also be *point covering*, that is, for each possible position of the object at least one intersection point of the grid lines lies inside the image of the object. (This condition is stronger. In the case of our sieve, it asks for every object to be touching at least one intersection point of the wires.) It is clear that for a fixed object size a point covering grid needs to have higher density than a line covering grid. For spherical objects, the grid densities are in a ratio of 1 to the square root of 2.

2.1 Calculation of the Grid Spacing

The construction of the vertical grid-lines is almost trivial, these can be obtained by considering lines that originate at the vanishing point (in the “middle” of the horizon) and meet a line at the footing of the agent with the required spacing (corresponding to the object size). Usually, this line will be outside the image. When traversing the lines, the loop statement starts at the intersection of the vertical scan-lines with the image borders. (Practically, this requires the examination of several cases, because the camera matrix may be rotated by more than 90°, and thus, the correct intersecting image border has to be found.)

The horizontal lines are not as easy to come by. Figure 1 illustrates how the spacing of the horizontal grid lines for a line covering grid can be obtained. Each line starting in the eye point is the tangent to two spheres. The intersection point of these two spheres in the image is the intersection between the tangent and the projection plane.

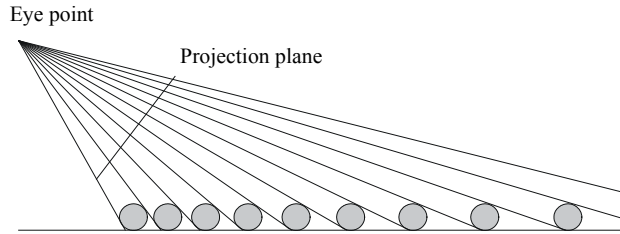


Fig. 1. Obtaining the horizontal grid lines for a line covering grid.

The horizon itself can be described by its distance to the center of the image and its rotation angle. The position and orientation of the horizon only depends on the camera rotation and not on the camera height. Usually the rotation of the camera is described with a rotation matrix C , providing a transformation from camera coordinates to world coordinates. Let φ be the angle between the optical axis of the camera and the ground, γ be the vertical opening angle of the camera, r be the vertical resolution of the image. Let x be an vector in camera coordinates parallel to the optical axis of the camera and $x'=Cx$ be that vector converted to real world coordinates. This leads to the following formula for the distance d between the horizon and the center of the image:

$$\varphi = \frac{r}{2} \frac{\sin\left(\operatorname{atan}\left(\frac{x'_3}{\sqrt{x'^2_1 + x'^2_2}}\right)\right)}{\sin\left(\frac{\gamma}{2}\right)}$$

where x'_3 is the vertical component of x' .

Let y and z be vectors in camera coordinates, pointing right and up and $y'=Cy$, $z'=Cz$ be the vectors converted to real world coordinates. Let v be the vector that

describes the direction of the intersection line of the ground plane and the plane defined by y' and z' . Then the angle of the horizon in the image is the angle between y' and v .

This provides the means to obtain a turned grid in the case the horizon is not parallel to the lower margin of the image.

As mentioned above, the grid becomes denser towards the horizon. Theoretically, the grid could end at a certain distance to the horizon that is determined by the maximum camera-to-object distance. For a practical robot application it can be necessary to continue the grid to the horizon and a little bit above it to avoid distant objects to “fall through” the grid if the orientation of the camera is not completely reliable.

When using the method for determining the structure of the grid presented above, it might happen that the one and only intersection between an object and a grid line is very close to the outline of the object. It is a good rule of thumb to choose the grid at least twice as small as optimal point coverage would require.

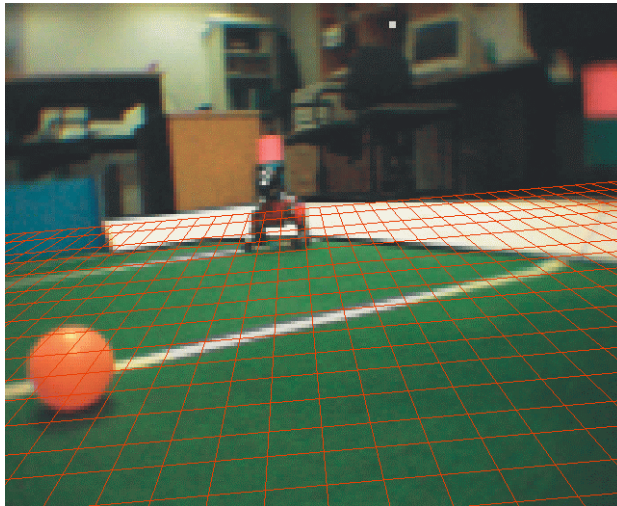


Fig. 2. Scan grid imposed on image for a given camera matrix

2.2 Scanning for Suspended Objects

Unfortunately, our grid only covers ground-based objects that stretch below the horizon. If there are objects of interest located at the ceiling, a similar grid may be considered, stretching from the opposite direction towards the horizon. Again, the grid spacing is determined by the size and shape of the objects in question, and the height of the ceiling above the camera position.

In our case, we are also scanning the image for landmarks (‘flags’), which here are cylindrical upright two-colored beacons, suspended on practically invisible sticks slightly above eye level. Because of the fact that nothing above the flags is of any interest to us and because of their vertical nature, we are scanning the image portion above the horizon with a different scheme:

- The horizontal grid remains as it is, i.e. it is narrow at the horizon and gets wider with growing angular difference.
- The vertical grid is not bent but upright, which is useful for the pattern matching described below, but increases the number of sample points.
- Only the lower portion of the area has to be sampled. The lower edge of flags happens to be almost aligned with the camera level and thus is to be found close to the horizon. If no grid-point near the horizon signifies that a flag-candidate has been found, the according vertical column above the area where flags should be seen can be abandoned.

We found that the camera parameters of our robot are not always interpolated in such a way as to allow a good estimate of the position of the horizon, because it uses four legs for locomotion and moves in a rolling, irregular gait. To solve this problem, we enhance the data that is gathered from the leg joint sensors with information from a tilt sensor. Together, they give a very accurate estimate of the horizon line.

3 Scanning the Grid Using a Pattern-Matching Method

If individual objects are robustly identified by their color class, the direction of the scanning of the grid is unimportant. However, it is often better to identify objects by the color properties of a number of adjacent grid-points. The field boundary, for instance, is characterized by a number of white sample points above several green samples (because the boundary is white and the floor conveniently happens to be green). Note that this is not true for every grid-point hitting the boundary, because in certain areas the boundary will invariably be partially occluded by other objects. Not every grid-point will return significant results, but fortunately, the field boundary tends to be rather large and thus typically yields numerous hits.

Other examples of objects that can nicely be found by comparing adjacent grid-points are the goals and other robots. However, in all cases the significant color-sequence will have to lie within the scan-direction, i.e. a vertical scan will fail to find vertically aligned field lines. Nonetheless, most objects we are interested in can be found with a vertical scan direction (that is, not vertical to the image, but following a grid-line that intersects the horizon). Currently, field lines are ignored, while later implementations might perform successive scans in several directions to find field-lines along with their orientation.

The scanning works as follows: the previous sample pixel is stored in a small ring buffer that makes a small number of preceding grid-points available. After each sampling, the sequences are matched against a set of signaling color-sequences (such as “green, green, white, white” for a field boundary, “orange” for a ball, “pink, blue” for the pink-blue flag, “green, green, yellow, yellow” for the yellow goal and so on). This implements a simple one-dimensional pattern-matching. It is possible to use multiple signaling sequences for the same object. The idea of using the sequences is to make the method more robust against random occurrences of colors (as they may happen due to color class mismatches or alien objects in the field of vision).

The positions found do of course not always contain the signaled article. This is checked by specialized identification routines, which also establish the boundaries of identified objects.

Every object class is handled by its respective ‘identifier’, i.e. there is a sub-routine for ball identification, one for the pink-blue flag, one for opponent robots and so on. On invocation, the identifier checks first whether an object of the corresponding class has already been discovered and the grid-point submitted by the pattern-matching mechanism lies within its outline (usually just its bounding box). If this is the case, the identification is cancelled (the object corresponding to the grid-position is known already). Because different objects can have intersecting boundaries, it is necessary to perform this test at the level of the identifier.

The identifier tries to find the actual outlines of the object (which may stretch over many grid-points) and if it succeeds, the object is stored in a list.

After the scanning is finished, it will be necessary to ensure the consistency of the list of identified objects, i.e. to cancel out multiple balls or to reject inconsistently identified flags. For instance, the consistency problem of multiple balls is solved by rejecting all but the “biggest” visible ball. (Consistency with current beliefs of the agent about its own position and the ball position are not handled at this stage of image processing.)

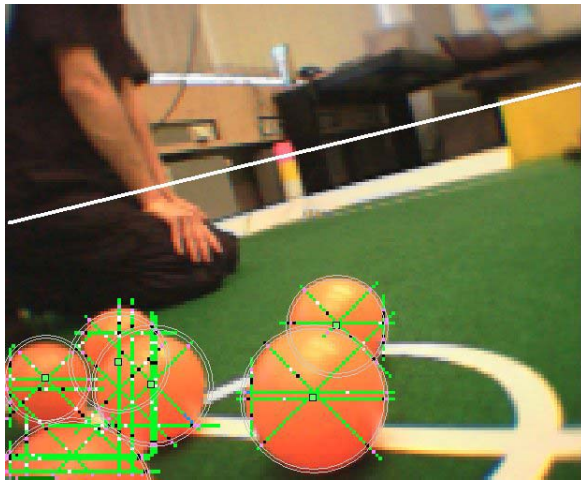


Fig. 3. Note that while in the game only a single ball occurs, in testing conditions we can have as many balls as we want.

4 Example of Object Identification: Identifying a Ball

A good example of a simple identification module for objects is our ball recognition:

We selected the simplest pattern matching strategy to find ball candidates. A single orange pixel was said to be a candidate for a ball. (This yields numerous erroneous pixels, however, we consider the ball to be too important to be missed.) The ball identifier works in two steps: The first step tries to find a point near the center of the visible part of the ball. The second step starts from the point obtained by step one to scan into eight directions for a transition from orange to green, white, sky-blue or yellow. If such a transition is found, the point where the transition takes place is marked as a candidate to lie on the outline of the ball. A transition from orange to another color stops the scanning process in that direction, and the point is not marked, because it is likely to signify a partly occluded edge of the ball.

In the current approach, we use a color lookup table to detect the color transitions. If there is a strong specular highlight within the ball this can cause a preliminary stop of the scanning process, which is a general fault of our absolute color classification scheme. Nonetheless, usually this has no influence on the ball-recognition

Out of all candidate points the triple that spans the triangle with the highest girth is determined. The circle defined by these three points describes the outline of the ball. Its radius can be used for distance calculation and its center point to calculate the angle to the ball.

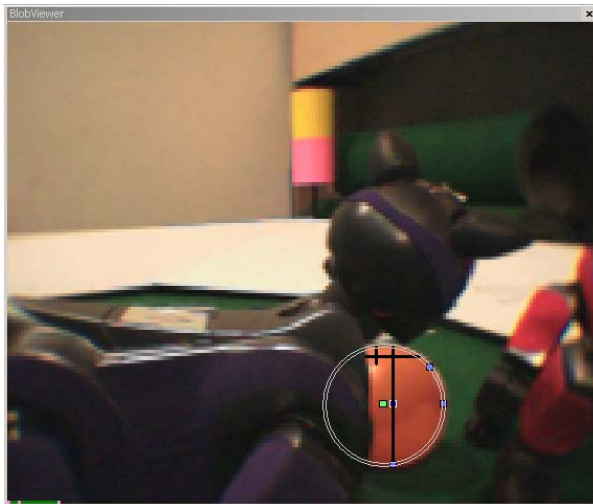


Fig. 4. Identification of a partially occluded ball

5 Results

In addition to the ball recognition module, similar identification modules for field markers, goals and other robots have been implemented. The complete processing of an image takes about 10ms on our robots, which enables us to process every frame (at a rate of 25 fps). Additionally, we can make use of the full camera resolution, rather than resorting to down-scaled versions of the frames. This was not possible with the

previous approach, where we segmented the image using runlength-encoding to generate irregular octogons of similar color. Our new method is also much more reliable, because the individual identifiers could be fine-tuned for the objects in question, which gives better results than evaluating the dimensions of generic octogons. Our robots have been successfully tested in this year's RoboCup world competition in Fukuoka, Japan.

In the future, we are planning to enhance the method to recognize additional markers on the field, especially field lines, so that the classical landmarks (flags) can become obsolete.

While the method has been tailored for soccer-playing robots, we consider it applicable in other situations, where robots have to move on flat, even surfaces and have to recognize recurring, isolated objects.

References:

RoboCup: Website of the RoboCup Federation, <http://www.robocup.org> (last visited August 2002)

Hajiaghayi, M.T.; Jamzad, M; et al.: A Fast Vision System for Middle Size Robots in RoboCup, The RoboCup 2001 International Symposium, RoboCup 2001, Springer, Heidelberg

Burkhard, H.-D. et al. : Team Report GT2001, RoboCup 2001, Springer, Heidelberg

Bruce, J., Balch, T., Veloso, M.: Fast and Inexpensive Color Image Segmentation, in Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00), Vol. 3, October, 2000, pp. 2061-2066

Kaelbling, L.P.; Oates, T., Hernandez, N., Finney, S.: Learning in Worlds with Objects, 2000, <http://www.cs.umass.edu/papers/kaelbling.pdf>

Kierzenkowski K., Puchalski J.: Object contour recognition using oriented grids. In Machine Graphics and Vision vol. 3, no. 4, 1994, pp. 657-665

Quek, F.K.H.: An algorithm for the rapid computation of boundaries of run length encoded regions. In Pattern Recognition Journal 2000, 33, pp. 1637-1649

Triesman, A.: Preattentive Processing in Vision, in Computer Vision, Graphics, and Image Processing 1985, 31, pp. 156-177